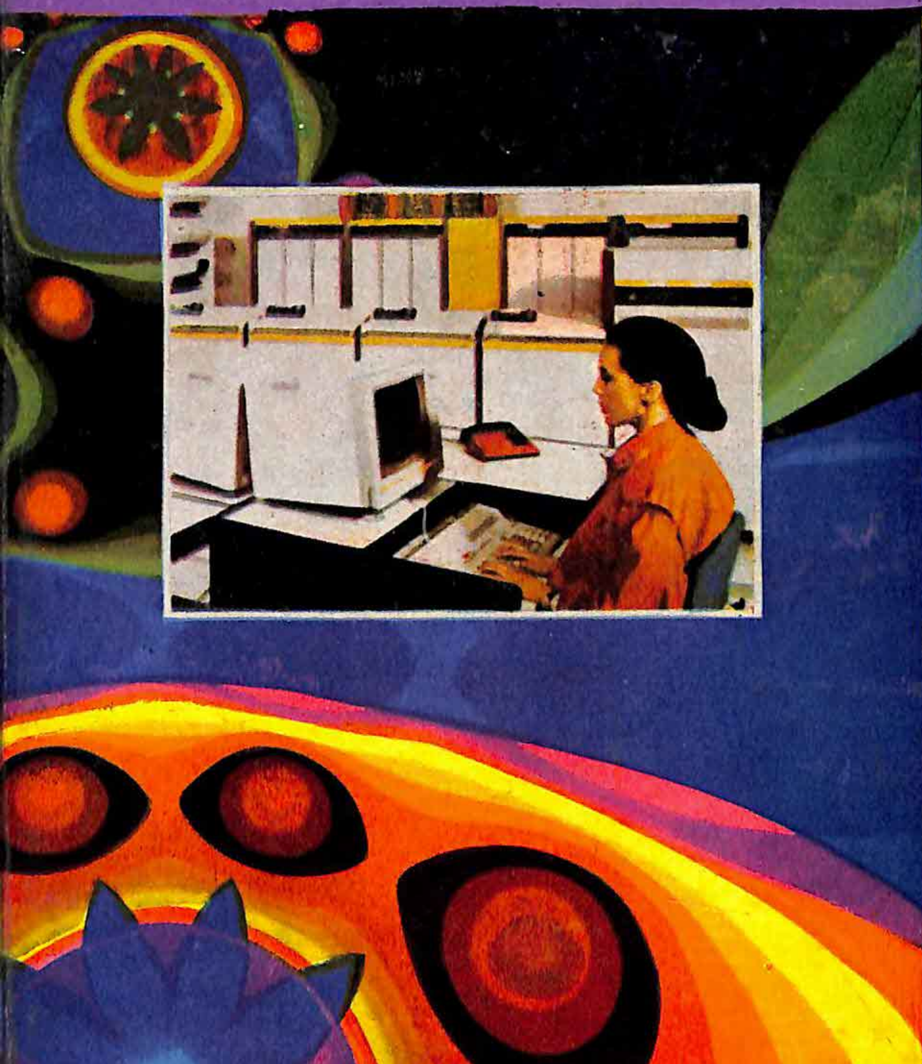
হাতে কলমে

কম্পিউটার ও বেসিক প্রোগ্রামিং



প্রমথেশ দাস ■ অনীশ দেব

হাতেকলমে

কম্পিউটার ও বেসিক প্রোগ্রামিং

প্রমথেশ দাস ■ অনীশ দেব

শৈব্যা গ্রন্থন বিভাগ ● ৮/১এ, শ্যামাচরণ দে স্ট্রীট, কলি-৭৩

Hate Kalame
Computer-O-BASIC Programming
by
Pramathes Das & Anish Deb

প্রথম প্রকাশ : কলকাতা বইমেলা, জানুয়ারী, ১৯৮৯

প্রকাশক :
দুলাল বল
৮/১এ, শ্যামাচরণ দে স্ট্রীট
কলিকাতা-৭৩

মুদ্রক :
লীলা ঘোষ
তাপসী প্রিণ্টার্স
৬, শিবু বিশ্বাস লেন
কলিকাতা-৬

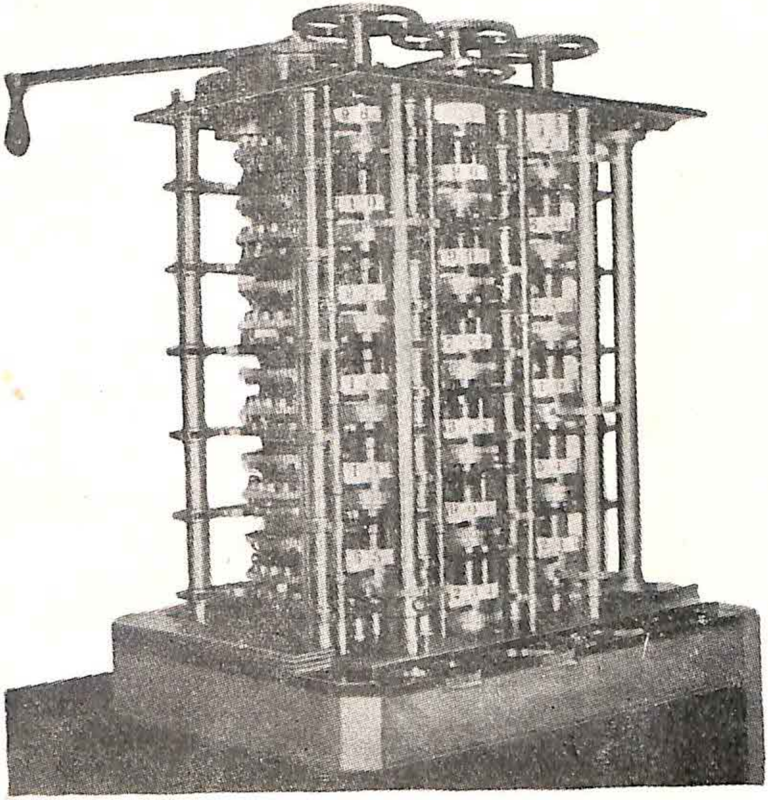
প্রচ্ছদ-চিত্র : কম্পিউটারে ফুটিয়ে তোলা
রঙিন নকশা ও পার্সোনাল
কম্পিউটারে কর্মরত কর্মী

দাম : পঁচিশ টাকা

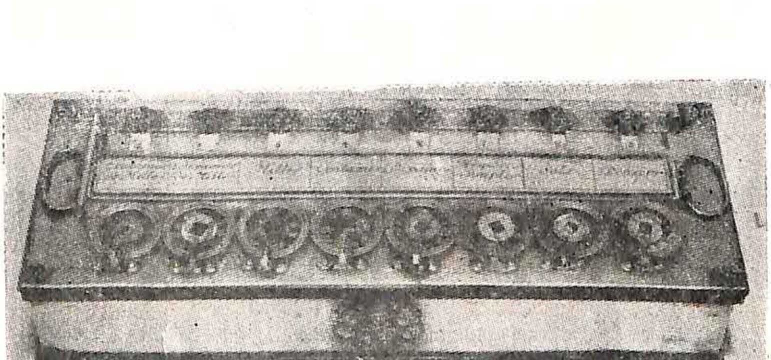
উৎসর্গ

শ্রীমন্মথরঞ্জন গুপ্ত ও
শ্রীজ্যোতিষচন্দ্র চক্রবর্তীকে

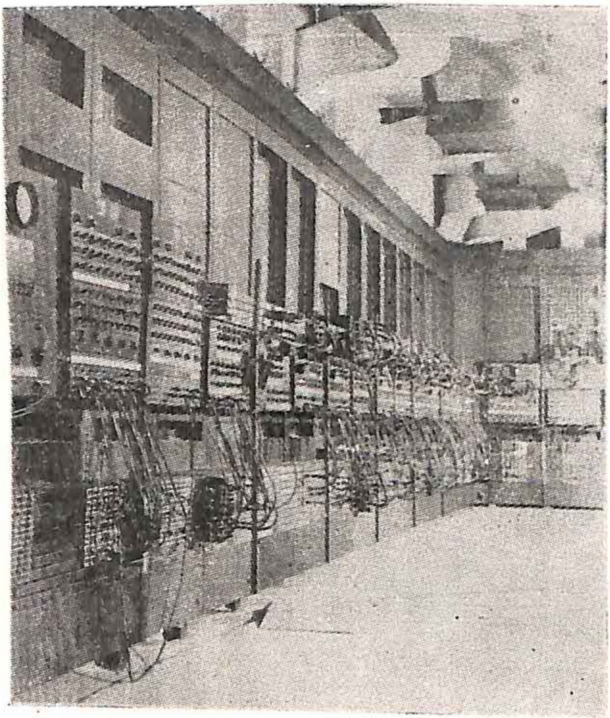
শ্রদ্ধেয় অধ্যাপক
ডক্টর তপনকুমার মুখোপাধ্যায়কে



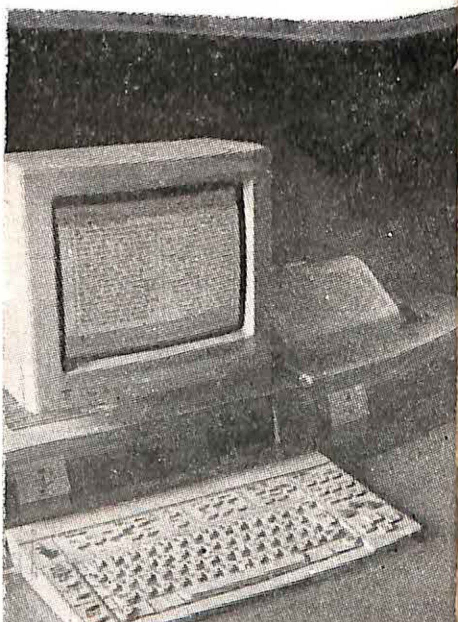
চার্লস ব্যাবেজের তৈরি প্রথম 'ডিফারেন্স ইঞ্জিন'। দ্বিতীয় 'ডিফারেন্স ইঞ্জিন' তৈরির পর ব্যাবেজ 'অ্যানালিটিক্যাল ইঞ্জিন' তৈরিতে মনোযোগ দেন।



পাস্কালের মেকানিকাল কম্পিউটার 'পাস্কালাইন'।



প্রথম ইলেকট্রনিক ডিজিটাল কম্পিউটার 'ইনিয়াক'-এর ছবি



আধুনিক পার্সোনাল কম্পিউটার।

□ এই বই প্রসঙ্গে

কম্পিউটার ও প্রোগ্রামিং আমাদের জীবনে ক্রমেই অপরিহার্য হয়ে উঠছে—বিশেষ করে ছাত্র-ছাত্রী, গবেষক বা অফিস-কর্মীদের কাছে। এই যন্ত্রটির আশ্চর্য ক্ষমতা সকলকেই মুগ্ধ করে—আমাদেরও করেছিল। সেই মুগ্ধতা থেকেই এই বই লেখার সূচনা। প্রথম লেখক এরকম একটি বইয়ের কথা ভেবে স্বাধীনভাবে লেখার কাজ শুরু করেছিলেন সুদূর জর্ডনে বসে। আর অনুজ লেখক একই চিন্তাধারা নিয়ে আলাদাভাবে কাজ শুরু করেছিলেন কলকাতায়। পরে যখন আমাদের মুখোমুখি দেখা হয়, কথা হয়, তখনই আমরা এই বই লেখার কাজটিকে যৌথ পরিকল্পনা হিসেবে গ্রহণ করি। এই পরিকল্পনা রূপায়ণের কাজে আমাদের পরিশ্রমের পাশাপাশি উল্লেখ করতে হয় প্রকাশক শ্রীদুলাল বলের ভূমিকা। আমাদের যে-কোনও অনুরোধকেই তিনি সম্মান দেখিয়েছেন এবং বইটির মুদ্রণ পারিপাট্যের জন্য ষোলো আনা চেষ্টা করেছেন। এরকম সহমর্মী প্রকাশক বিরল। তাঁকে কৃতজ্ঞতা জানাই।

বইটিতে পঞ্চাশটিরও বেশি বেসিক প্রোগ্রামের উদাহরণ রয়েছে, এবং সবক'টি প্রোগ্রামই কম্পিউটারে চালানো। এর জন্য মূলত তিনটি কম্পিউটার আমরা ব্যবহার করেছি—সিনক্লেয়ার পিসি, আই. ডি. এম. পিসি এবং ক্যাসিও পিবি—770। বইটি ব্যবহার করে কোনও ব্যবহারকারী যাতে নিজে নিজেই বেসিক প্রোগ্রাম শিখতে পারে সেইদিকে নজর রেখেই লেখার চেষ্টা করেছি। আমাদের চেষ্টা সফল হলেই আমাদের পরিশ্রমের একটা অর্থ খুঁজে পাওয়া যাবে।

পরিশেষে একটা কথা : বইয়ের লেখার কাজ যেমন আমরা দুজনে সমান-ভাবে ভাগ করে নিয়েছি তেমনই এর ভুল-ত্রুটির দায়িত্বও আমাদের। এই বইতে মুদ্রণগত, তত্ত্বগত বা তথ্যগত কোনও ত্রুটি যদি কারও নজরে পড়ে তাহলে তা আমাদের জানালে বাধিত হব।

প্রমথেশ দাস
অনীশ দেব

কলকাতা বইমেলা, ১৯৮৯

□ সূচীপত্র □

হাতেকলমে সিরিজের অন্যান্য বই

হাতেকলমে ইলেক্ট্রনিক্স রত্নেশ্বর রায়	১২·০০
হাতেকলমে পদার্থবিজ্ঞান অজয় চক্রবর্তী	১২·০০
হাতেকলমে রসায়ন কমল চক্রবর্তী	১২·০০
হাতেকলমে জীবন বিজ্ঞান সন্দীপ সেন	১২·০০
হাতেকলমে গণিত অরূপরতন ভট্টাচার্য	১২·০০
ইলেক্ট্রনিক্স ও ইলেক্ট্রনিক্স প্রোজেক্ট রত্নেশ্বর রায়	১২·০০

১ □ কম্পিউটার কেমন করে এল

সবচেয়ে পুরোনো কম্পিউটার

এক-দুই কেমন করে গুনতে হয় সেটা সবাই জানে। যে কোনও ছোট্ট ছেলেও এক থেকে একশো পর্যন্ত গুনে ফেলবে অনায়াসে। চাই কি একশো পেরিয়ে হাজারের দিকেও পাড়ি জমাতে পারে। এখন একটু ভেবে দেখা যাক, এই সংখ্যা গোনার ব্যাপারে কোন্ কোন্ জিনিসের সাহায্য নিই আমরা। এক, গলার স্বর—যা সংখ্যাগুলো শব্দ করে উচ্চারণ করে। আর কখনও কখনও দু-হাতের দশ আঙুল।

ছোটবেলায় অঙ্ক শেখার শুরুতে আমরা আঙুলে কর গুনে ধাপে ধাপে এগোই। যখন গুনতে গুনতে দু-হাতের দশটা আঙুলই খরচ হয়ে যায়, তখন আবার প্রথম আঙুল থেকে গুনতে শুরু করি। সুতরাং অনায়াসে একথা বলতে পারি, মানুষের হাতের আঙুলই হল সবচেয়ে প্রাচীন কম্পিউটার।

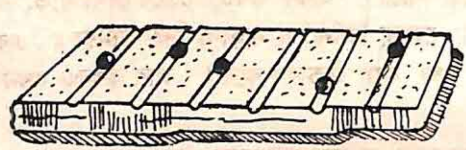
হাতের আঙুলের সংখ্যা দশ হওয়ার ফলেই হয়ত এক থেকে নয়, ও শূন্য, মোট এই দশটি সংখ্যার চলন হয়েছিল আমাদের এই ভারতবর্ষে। এই সংখ্যা পদ্ধতি আজও সারা পৃথিবীর মানুষ ব্যবহার করে চলেছে—যার নাম হল দশমিক সংখ্যা পদ্ধতি বা ডেসিম্যাল নাম্বার সিস্টেম।

যদি এমন হত যে পাঁচ-এর বেশি আর কোনও সংখ্যা নেই, তাহলে কোনও জিনিসের সংখ্যা পাঁচ পেরোলেই আমাদের বলতে হত 'অনেক' বা 'অসংখ্য'। সত্যি সত্যি এমনটা হয়েছিল আফ্রিকার হটেনটট্ উপজাতিদের ক্ষেত্রে। ওরা তিন-এর বেশি মোটেই গুনতে পারত না। সবচেয়ে আধুনিক যে কম্পিউটার মানুষের হাতে রয়েছে, সে কিন্তু হটেনটটেরও এককাঠি ওপরে। এই কম্পিউটার বেচারা জানে শুধু শূন্য এবং এক, এছাড়া আর কোনও সংখ্যাই চেনে না। তবু সে কি করে অঙ্কে বিশারদ হয়ে উঠেছে সেটা আমরা পরে দেখতে পাব।

দশ পর্যন্ত গুনতে শিখে মানুষের অসুবিধে হয়নি। দিব্যি হাতের আঙুলে কাজ চলছিল। তারপর একদিন একজনের মাথায় একটা বুদ্ধি খেলে গেল। আঙুলের বদলে ছোট ছোট নুড়ি ব্যবহার করলে কেমন হয়! এই বুদ্ধি খেলতেও সময় লেগেছিল তিরিশ-চল্লিশ লক্ষ বছর। সুতরাং তখন থেকেই

দশটা নুড়ি বা সে-জাতীয় অন্য কিছু কোনও কৌটোয় ভরে মানুষ সব সময় হাতের কাছে রাখতে লাগল। আর প্রয়োজন হলেই এক থেকে দশ পর্যন্ত চটপট গুনে ফেলতে লাগল। কিন্তু এই নুড়ি রাখার কৌটোকে কোন্ চেহারা দিলে যে সহজে হিসেব-নিকেশ করা যাবে সেটা ভেবে বের করতে প্রস্তরযুগের পণ্ডিতদের বহু শতাব্দী লেগে গিয়েছিল।

আজ থেকে প্রায় পাঁচ হাজার বছর আগে টাইগ্রিস-ইউফ্রেতিস উপত্যকায় যে সভ্যতা জেগে উঠেছিল, সেখানকার বুদ্ধিমান মানুষরাই আবিষ্কার করল পৃথিবীর প্রথম যন্ত্রগণক বা ক্যালকুলেটর। একটা মাটির ফলক তৈরি করল তারা। তার চেহারা অনেকটা আমাদের চেনা স্লেট-এর মতো—তবে অনেক বেশি পুরু। তার মধ্যে লম্বা লম্বা গর্ত কেটে নিল সমান্তরালভাবে। তারপর সেই গর্তের ওপরে বসিয়ে দিল অঙ্ক কষার নুড়িগুলো। আসল হিসেবের সময় নুড়িগুলোকে গর্ত বরাবর এ-মাথা থেকে ও-মাথায় চালিয়ে চটপট মিলে যেত অঙ্কের উত্তর। তাছাড়া এর সাহায্যে এক হাতেই সব কাজ সারা হয়ে যায়, অন্য হাতটা খালি থাকে। ইচ্ছে করলে খালি হাতটা অন্য কোনও কাজেও লাগানো যায়। এই সব সুবিধেগুলো তখনকার মানুষদের মুগ্ধ করে তুলেছিল প্রাগৈতিহাসিক কম্পিউটারটির প্রতি। আজকের দিনে একথা ভাবলে হয়ত আমাদের অবাক লাগবে।



পৃথিবীর প্রথম যন্ত্রগণক

এরপর খ্রীস্টপূর্ব পঞ্চম শতাব্দীতে মিশরের মানুষ এই যন্ত্রগণকের ব্যবহার শিখল। অবশ্য তাদের প্রায় একশো-দেড়শো বছর আগেই চীনদেশ, প্রাচীন গ্রীস ও রোমে এর ব্যবহারের কথা জানা গেছে। বিশেষ করে গ্রীকরা এই যন্ত্রকে কলকব্জা দিয়ে বেশ উন্নত করে তুলেছিল। তার প্রমাণ পাওয়া গেছে 1901 সালে। গ্রীস দেশের একটি দ্বীপের নাম অ্যান্টিকিথেরা। তারই কাছাকাছি কোনও এক সময়ে ডুবে গিয়েছিল একটা জাহাজ। 1901 সালে উদ্ধার করা হল সেই জাহাজের ধ্বংসাবশেষ। সেখান থেকে পাওয়া গেল কলকব্জা দিয়ে তৈরি একটি যন্ত্রগণক। পরীক্ষায় জানা গেল, যন্ত্রটা আনুমানিক খ্রীস্টপূর্ব 82 সালে তৈরি! ছোট-বড় নানান্ চাকতি জুড়ে তৈরি এই যন্ত্রটিকে বেশ জটিল 'কম্পিউটার'ই বলতে হবে। পরে আরও জানা গেছে, ওই চাকতিগুলো নাকি সূর্য, চন্দ্র, বিভিন্ন গ্রহ ও কিছু কিছু নক্ষত্রের গতিবিধি নির্দেশ করছে। এবং ওই চাকতিগুলোর সাহায্যে জ্যোতির্বিদ্যা-সংক্রান্ত গণনা করা সম্ভব।

আর্কিমিডিস-এর সময়ে, অর্থাৎ খ্রীস্টপূর্ব তৃতীয় শতাব্দীতে, এক ধরনের 'কম্পিউটার' চালু ছিল। তাকে বলা হত 'অ্যাবাকাস'। জিনিসটা আর কিছুই না, নেহাৎ একটা কাঠের ট্রে-র ওপরে কিছুটা কালচে বালির স্তর ছড়ানো থাকত। সেই বালিতে আঙুল বা ছুঁচলো কিছু দিয়ে দাগ টেনে অঙ্ক কষত সবাই। ঠিক যেন আজকের স্লেট-পেন্সিলের মতো। এক সৈনিকের হাতে নিহত হওয়ার মুহূর্তে এই রকমই একটা বালির 'স্লেটে' জ্যামিতিচর্চা করছিলেন আর্কিমিডিস।

খ্রীষ্টপূর্ব 212 সালে রোমানরা গ্রীস দেশ আক্রমণ করল। তারা হয়ত সহজেই গ্রীকদের যুদ্ধে হারিয়ে দিতে পারত, যদি না আর্কিমিডিস নিজের বৈজ্ঞানিক বুদ্ধি খাটিয়ে নানান কলাকৌশল করে রোমান সৈন্যদের বাধা দিতেন। অবশেষে রোমানরা যখন সাইরাকিউস অধিকার করে ফেলল, তখন চারপাশের বিপদ ও গোলমালের কথা ভুলে আর্কিমিডিস অ্যাবাকাসে মাথা ঝুঁকিয়ে জ্যামিতি নিয়ে ভীষণ ব্যস্ত।

রোমান সেনাপতি মার্সেলাস আর্কিমিডিসের আশ্চর্য প্রতিভার কথা জানতেন। তিনি তাঁর সৈন্যদের বলেছিলেন আর্কিমিডিসকে যেন জীবিত অবস্থায় উপযুক্ত সম্মানের সঙ্গে তাঁর কাছে নিয়ে আসা হয়। কিন্তু যে রোমান সৈনিকটি আর্কিমিডিসের কাছে হাজির হয়ে তাঁকে বলে, 'শিগগির আমার সঙ্গে চলুন। সেনাপতির আদেশ।', তার বোধহয় ধৈর্য কম ছিল। কারণ তার আদেশের উত্তরে অ্যাবাকাস থেকে মাথা না তুলেই হাত নেড়ে ইশারা করেছিলেন আর্কিমিডিস, বলেছিলেন, 'আঁকা বৃত্তগুলোকে নষ্ট কোর না।' বালির ওপরে তখন কয়েকটা বৃত্ত আঁকা ছিল। কিন্তু সৈনিকটি অতশত না বুঝেই এক কোপে শেষ করে দিয়েছিল আর্কিমিডিসকে।

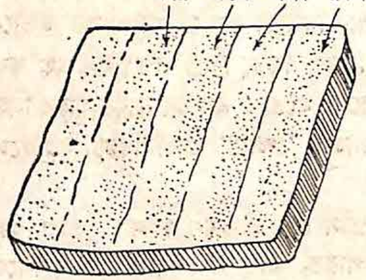
এই ঘটনা জানতে পেরে মার্সেলাস অত্যন্ত দুঃখ প্রকাশ করেন এবং আর্কি-মিডিসের জন্য রাজকীয় সমাধির আয়োজন করেন।

বালির এই স্লেটের নাম 'অ্যাবাকাস' হওয়ার পেছনে দুটো কারণ থাকতে পারে। প্রথমত, গ্রীক ভাষায় 'অ্যাবাক্স' শব্দের অর্থ চ্যাপ্টা কাঠের বোর্ড বা পায়াহীন টেবিল। আর দ্বিতীয়ত, হিব্রু ভাষায় 'অ্যাবাক' বলা হত ধুলোকে —অর্থাৎ, অ্যাবাকাসের ক্ষেত্রে, বালিকে।

মাটির স্লেটে নুড়ি সরিয়ে হিসেব করার যে যন্ত্রের কথা আগে বলেছি, সেটাই সামান্য রূপবদল করে ছড়িয়ে পড়ল পৃথিবীর নানান জায়গায়। তবে যন্ত্রটি ছড়িয়ে পড়তে সময় লেগেছিল কয়েক হাজার বছর।

খ্রীস্টপূর্ব চতুর্থ শতাব্দীতে যে অ্যাবাকাসের চলন ছিল, তাকে বলা যায় গণনা-ফলক। কারণ, এই অ্যাবাকাসে একটা ফলকের ওপর বেশ কয়েকটা সমান্তরাল রেখা টানা হত। এর এক-একটা রেখা এক-এক রকম মান নির্দেশ করত।

অর্থাৎ, একক, দশক, শতক, ইত্যাদির ঘরের মতো। কখনও এই দাগ টানা হত কাঠের ওপরে, কখনও বা পার্চমেণ্ট কাগজে আঁকা হত, কখনও পাথরে খোদাই করা হত, এমন কি কাপড়ের ওপরে সেলাই করেও আঁকা হত। এই জিনিসটাকে গ্রীকরা বলত 'অ্যাবাকিয়ন', রোমানরা বলত 'অ্যাবাকাস'। সমান্তরাল রেখাগুলোর ওপরে নুড়ি বসিয়ে যত রকম হিসেব-নিকেশ করা হত। নুড়িকে ল্যাটিন ভাষায় বলা হয় 'ক্যালকুলাস', তার থেকেই হয়ত ইংরেজি 'ক্যালকুলেট' বা 'অঙ্ক কষা' কথাটার জন্ম হয়েছে।

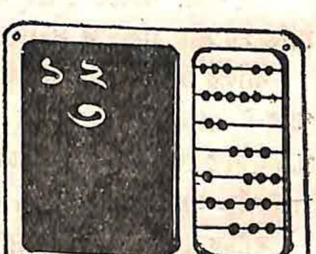


গণনা-ফলক

এ-জাতীয় গণনা-ফলকের ছবি অনেক পাওয়া গেলেও একটি মাত্র গ্রীক গণনা-ফলক মানুষ হাতে পেয়েছে। পনেরো সেণ্টিমিটার লম্বা সাড়ে বারো সেণ্টিমিটার চওড়া পাথরের একটা আয়তাকার ফলক পাওয়া গেছে সালামিস দ্বীপে।

সাধারণত অ্যাবাকাস বলতে আমরা যে জিনিসটাকে বুঝি তার চেহারা গণনা-ফলকের চেয়ে কিছুটা উন্নত। যেমন, নুড়িগুলো হয়ত ফুটো করে কোনও তার কিংবা রডের মধ্যে পুঁতির মালার মতো পরিয়ে তারপর সহজে নাড়াচাড়া করানো হয়। এর ফলে অঙ্ক কষার গতিও অনেক বেড়ে যায়।

ছোটবেলায় একধরনের স্লেট আমরা অনেকেই ব্যবহার করেছি, যার এক পাশে থাকত কয়েকটা সমান্তরাল তার। সেই তারে পরানো থাকত ছোট ছোট রঙিন পুঁতি—যা দিয়ে আমরা সহজে এক-দুই গুণতে শিখি। স্লেটের অংশটুকু বাদ দিয়ে শুধু পুঁতির অংশটুকু রাখলে যেরকম চেহারা দাঁড়ায় তা অনেকটা অ্যাবাকাসেরই মতো।



ছোটবেলার রঙিন পুঁতিওয়ালা স্লেট

রোমানদের মধ্যে কয়েক রকম অ্যাবাকাসের চলন ছিল। তার মধ্যে সবচেয়ে অবাক করে সপ্তদশ শতাব্দীতে ইটালিতে যে ব্রোঞ্জের অ্যাবাকাস ব্যবহার করা হত সেটা। কারণ তার সঙ্গে আজকের জাপানী অ্যাবাকাসের কাঠামো বেশ মিলে যায়। এইরকমই একটি অ্যাবাকাসের নমুনা বর্তমানে ব্রিটিশ মিউজিয়ামে রয়েছে।

ষোড়শ শতাব্দী নাগাদ সম্ভবত চীনদেশ থেকেই অ্যাবাকাসের ব্যবহার ছড়িয়ে পড়ে জাপানে। তবে জাপানীরা এর নাম দিয়েছিল 'সোরোবান'।

রোমান বনাম ভারতীয় সংখ্যা পদ্ধতি

প্রায় পনেরো শো বছর ধরে গ্রীক, রোমান ও পরে মধ্যযুগের ইওরোপীয়রা এমন সব যন্ত্রগণক নিয়ে কাজ করেছে যাতে একটা নুড়ি বা পুঁতি কোন্ জায়গায় রয়েছে তার ওপরে নির্ভর করে সংখ্যাটির মান। একে বলা হয় কোনও সংখ্যার স্থানীয় মান। যেমন 31 ও 13 মোটেই এক সংখ্যা নয়। অর্থাৎ, নুড়ি দিয়ে 3 ও 1 এর জায়গা পালটা-পালটি করে দিলেই সংখ্যার মান বদলে যায়। সুতরাং, একক, দশক ইত্যাদি স্থানের ওপরে নির্ভর করছে 3 ও 1, এই সংখ্যা দুটির মান কত। যখন 13 লিখছি, তখন 1 এর অর্থ 10, কারণ 1 সংখ্যাটি বসে আছে দশকের জায়গায়।

অতএব, $13 = 10 + 3$
আবার একই নিয়মে,
$31 = 30 + 1$

সুতরাং এই নিয়ম থেকে বলা যায়, প্রত্যেকটি সংখ্যার মান স্থান-নির্ভর।

গ্রীক-রোমানরা আরও একটি জিনিস করত। শূন্য বোঝাতে যন্ত্রের সমান্তরাল রেখা বা গর্তের একটাকে খালি রেখে দিত। কিন্তু ওরাই যখন আবার এইসব যান্ত্রিক পদ্ধতি রেখে হাতে লিখে হিসেব কষত, তখন রোমান সংখ্যা-পদ্ধতি ব্যবহার করত, যাতে স্থান-নির্ভর মান বা শূন্যের কোনও চলন ছিল না। যেমন, রোমান হরফে, M এর অর্থ 1000, L এর অর্থ 50, X এর অর্থ 10 এবং V এর অর্থ 5। এখনও আমরা দেওয়াল ঘড়িতে বা কোন্ ক্লাসে পড়ি সেটা লিখতে মাঝে মাঝে রোমান সংখ্যা ব্যবহার করে থাকি।

রোমান হরফে অঙ্ক কষার ফলে সময় অনেক বেশি লাগত, হিসেবটাও হয়ে দাঁড়াত জট পাকানো, কঠিন। কিন্তু তবুও ওরা 'শূন্য' সংখ্যাটার প্রয়োজন বুঝতে পারেনি।

ভারতবাসীরা এই সমস্যার সমাধান করে ফেলেছিল বহু আগেই। দশমিক সংখ্যা পদ্ধতি, শূন্যের ব্যবহার, সবই তারা জেনে গিয়েছিল, এবং অঙ্কও কষত সেই নিয়মে। একজন আরব অঙ্কবিদ, অল্-খোয়ারিজ্মি, ভারত থেকে এই সংখ্যার ব্যবহার আয়ত্ত করেন।

খোয়ারিজ্মি দশমিক সংখ্যা পদ্ধতি নিয়ে আরবীয় ভাষায় 'ইল্ম্ অল-জাবর ওয়াল্ মুকাবলা' নামে একটি বই লেখেন। এই বইয়ের নাম থেকেই পরে

'অ্যালজেব্রা' বা 'বীজগণিত' কথাটির উৎপত্তি হয়। সেই সময়টা নবম শতাব্দী হলেও অনুবাদের মাধ্যমে বিজ্ঞানচর্চার আদান-প্রদানের কোনও ঘাটতি ছিল না। ফরাসী পণ্ডিত জেয়ারবেয়ার অঙ্কের হিসেব-নিকেশের সুবিধের কথা ভেবে ইওরোপে অ্যাবাকাসের ব্যবহার নতুন করে শুরু করেন। এই হিসেবের জন্য তিনি বেছে নিয়েছিলেন অল্-খোয়ারিজ্মির দশমিক পদ্ধতি। তবে 0 এর ব্যবহার তিনি কি কারণে বাদ দিয়েছিলেন তা জানা যায়নি।

জেয়ারবেয়ার ধর্মযাজক থেকে ধীরে ধীরে পোপের আসনে অভিষিক্ত হন 999 সালে। বিভিন্ন বিষয়ে তাঁর জ্ঞান এত বেশি ছিল যে সবাই ভাবত জেয়ারবেয়ার জাদু জানেন। বিভিন্ন যন্ত্রপাতি তৈরিতে তাঁর দক্ষতা ছিল চমকে দেবার মতো। বলা যেতে পারে, ইওরোপীয় বিদ্যাচর্চার পুনর্জন্ম জেয়ারবেয়ারের সময় থেকেই শুরু হয়েছিল।

কিন্তু দশমিক পদ্ধতির জোরালো চর্চা শুরু হতে আরও কিছুদিন সময় লেগে যায়। অবশেষে খোয়ারিজ্মির তত্ত্ব ল্যাটিন ভাষায় অনুবাদ হয়ে প্রচারিত হল ইওরোপে। প্রচার যিনি করেছিলেন তাঁর নাম লিওনার্দো ফিবোনাচ্চি।

ইটালির পিসা নগরে বাস করতেন বলেই হয়ত ফিবোনাচ্চিকে 'পিসার লিওনার্দো' বলা হত। ফিবোনাচ্চির শিক্ষক ছিলেন মুসলমান। প্রথমে আলজিরিয়া ও পরে উত্তর আফ্রিকার বহু জায়গায় ঘুরে বেড়িয়েছেন ফিবোনাচ্চি। এইভাবেই আরবী ভাষা তিনি আয়ত্ত করেন। ফলে খোয়ারিজ্মির বইয়ের সঙ্গে তাঁর পরিচয় ঘটামাত্রই দশমিক সংখ্যা পদ্ধতির গুরুত্ব বুঝতে তাঁর কোনও অসুবিধে হয়নি। অবশেষে 1202 সালে প্রকাশিত হল ফিবোনাচ্চির বই 'লিবার অ্যাবাসি' বা 'অ্যাবাকাসের বই'। খোয়ারিজ্মি কোথা থেকে দশমিক সংখ্যা পদ্ধতির খোঁজ পেয়েছেন সেটা না জানার ফলে আরবী থেকে ল্যাটিনে যখন তাঁর তত্ত্ব অনুবাদ হয়, তখন এই সংখ্যা পদ্ধতিকে সবাই 'আরবীয় সংখ্যা পদ্ধতি' আখ্যা দেয়। কোনও কোনও ক্ষেত্রে অবশ্য 'হিন্দু-আরবীয় সংখ্যা পদ্ধতি'ও বলা হয়ে থাকে।

অঙ্কবিদ ফিবোনাচ্চি তাঁর এই বইতেই প্রথম বুঝিয়ে বললেন কোনও সংখ্যার স্থান-নির্ভর মানের কথা। যেমন, 213, 312, 231, 123, 132 ও 321 এর মান বিভিন্ন—যদিও ছ'টি সংখ্যাই 1, 2 ও 3 সংখ্যা তিনটি দিয়ে তৈরি। তাছাড়া 0 এর ব্যবহারও তিনি বিশদভাবে ব্যাখ্যা করেন। 'অ্যাবাকাসের বই' নাম হলেও আসলে ফিবোনাচ্চির বইতে পাটিগণিত ও বীজগণিত নিয়ে নানান হিসেব-নিকেশ ছিল। কারণ তখনকার দিনের ব্যবসায়ীদের জটিল হিসেব সহজ করে দেওয়ার চেষ্টাতেই বলতে গেলে ফিবোনাচ্চি বইটি লিখেছিলেন।

রোমান সংখ্যা পদ্ধতি ও দশমিক সংখ্যা পদ্ধতি নিয়ে নানান রকম মতভেদ দেখা দিল। অবশেষে ভারতীয় দশমিক পদ্ধতিরই জয় হল। কিন্তু তার

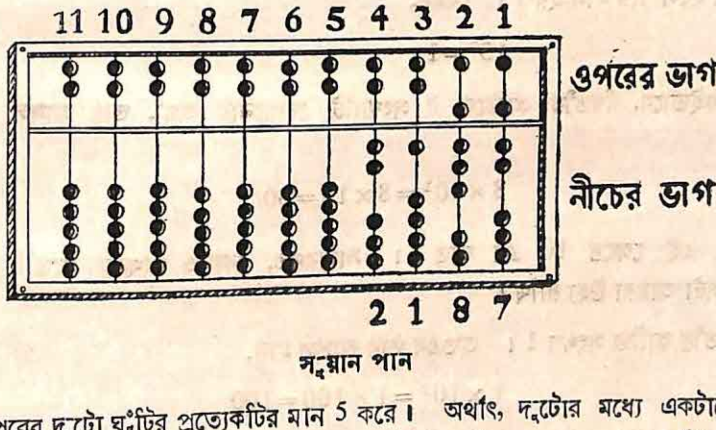
আগে ইওরোপের কোনও কোনও দেশে দশমিক পদ্ধতিতে অঙ্ক কষা আইন করে নিষেধ করে দেওয়া হয়েছিল। কিন্তু তবুও লোকে লুকিয়ে লুকিয়ে এই নিয়মে অঙ্ক করত। অবশেষে ষোড়শ শতাব্দীতে স্বীকৃতি পেল দশমিক নিয়ম। এছাড়া, দশটা সংখ্যার চেহারা কেমন হবে সেটাও মোটামুটি ঠিক হয়ে গেল। তখন দশমিক নিয়ম ছড়িয়ে পড়ল সারা পৃথিবীতে। আর অ্যাবাকাসে অঙ্ক কষার কাজেও ব্যবহার হতে লাগল দশমিক সংখ্যা পদ্ধতি।

কিন্তু কালক্রমে ইওরোপ ও ইংল্যান্ড থেকে ধীরে ধীরে অ্যাবাকাস বাতিল হয়ে গেল। বরং প্রাচ্যের কয়েকটি দেশ ও রাশিয়ায় এখনও এর চলন রয়েছে। যে-তিন ধরনের অ্যাবাকাস বর্তমানে দেখা যায়, তা হল চীন দেশের 'সুয়ান পান', জাপানের 'সোরোবান', আর রাশিয়ার 'স্কোটি'।

চীনের অ্যাবাকাস ঃ সুয়ান পান

চীনে ভাষায় 'সুয়ান' শব্দটির অর্থ 'ক্যালকুলেট' বা 'অঙ্ক কষা'। 'সুয়ান পান' শুধু চীন নয়, কোরিয়াতেও ব্যবহার করা হয়। এই যন্ত্রের 'পুঁতি'গুলো দেখতে অনেকটা ক্যারম খেলার ঘুঁটির মতো। তবে তার ধারগুলো ঘষে মসৃণ করে দেওয়া। কাঠের এই পুঁতিগুলো চলাফেরা করে বাঁশের কাঠির মধ্যে—অনেকটা যে ধরনের কাঠি দিয়ে চীনারা খেয়ে থাকে। আর গোটা কম্পিউটারটা একটা কাঠের দন্ড দিয়ে আড়াআড়িভাবে দু-ভাগে ভাগ করা।

'সুয়ান পান'-এ বাঁশের কাঠির সংখ্যা মোট এগারো। প্রতিটি বাঁশের কাঠিতে মোট সাতটা করে 'পুঁতি' থাকে ঃ ওপরের ভাগে দুটো, আর নিচের ভাগে পাঁচটা।



সুয়ান পান

ওপরের দুটো ঘুঁটির প্রত্যেকটির মান 5 করে। অর্থাৎ, দুটোর মধ্যে একটাকে সরিয়ে নিচের দিকে নিয়ে এলেই 5 সংখ্যাটি নির্দেশ করা হবে। আর নিচের ভাগের পাঁচটা ঘুঁটির প্রত্যেকটির মান 1। অর্থাৎ, যতগুলো ঘুঁটি সরিয়ে ওপরের

দিকে তোলা হবে, সেই সংখ্যাই নির্দেশ করবে যন্ত্রগণক। যেমন, ছবিতে একেবারে ডান দিকের কাঠিতে 7 সংখ্যাটি নির্দেশ করা রয়েছে। ওপরের ভাগের একটা 'পুঁতি' নিচে সরানো, অর্থাৎ, 5। আর এদিকে নিচের ভাগে দুটো ঘুঁটি ওপরে তোলা রয়েছে, অর্থাৎ, 2। সুতরাং নির্দেশ করা সংখ্যার মান,

$$5+2=7$$

এবারে খাড়াভাবে সাজানো এগারোটা বাঁশের কাঠির প্রসঙ্গে আসা যাক। আগে যেমন বলা হয়েছে, এক-একটা কাঠি ডান দিক থেকে যথাক্রমে একক, দশক, শতক, ইত্যাদি ঘর নির্দেশ করছে। ছবিতে 'সুয়ান পান'-এ 2187 সংখ্যাটি দেখানো রয়েছে। 2187 সংখ্যাটিতে প্রত্যেকটি অঙ্কের মান—অর্থাৎ, 2, 1, 8 ও 7—স্থান নির্ভর। 2187 কে পর পর লিখে আমরা বলতে চাইছি, দু-হাজার একশো সাতাশি। যদি একটু ভেঙে লিখি তাহলে,

$$2187 = \underset{\text{সহস্র}}{2000} + \underset{\text{শতক}}{100} + \underset{\text{দশক}}{80} + \underset{\text{একক}}{7}$$

এ থেকেই বোঝা যাচ্ছে, একক থেকে দশকের ঘরে বা দশক থেকে শতকে যেতে হলে কোনও সংখ্যাকে 10 এর গুণিতকে বাড়িয়ে যেতে হবে।

ডান প্রান্তের প্রথম কাঠিতে 7 সংখ্যাটি দেখানোর অর্থ তার মান,

$$7\times10^0=7\times1=7$$

এখানে 10^0 এর অর্থ দশের ঘাত শূন্য। যে কোনও সংখ্যার ঘাত শূন্য বসানো হলে তার মান হয় 1। যেমন,

$$13^0=1$$

একইভাবে, দ্বিতীয় কাঠিতে 8 সংখ্যাটি দেখানোর অর্থ, তার আসল মান,

$$8\times10^1=8\times10=80$$

অর্থাৎ, এই ক্ষেত্রে 10 এর ঘাত 1। সাধারণত, কোনও সংখ্যার ঘাত 1 হলে সেটা আমরা উহ্য রাখি।

তৃতীয় কাঠির সংখ্যা 1। অতএব তার আসল মান,

$$1\times10^2=1\times100=100$$

এখানে, 10^2 এর অর্থ, 10 কে 10 দিয়ে গুণ, বা দুটো 10 কে পরস্পর গুণ করা।

শেষ ও চতুর্থ কাঠিতে দেখানো রয়েছে 2। অতএব তার প্রকৃত অর্থ,

$$2 \times 10^3 = 2 \times 1000 = 2000$$

এখানে 10^3 এর অর্থ,

$$10 \times 10 \times 10 = 1000$$

আর চারটি ছাড়া বাকি সব কাঠিতেই দেখানো রয়েছে 0 সংখ্যাটি।

এইভাবে যদি আমরা বাঁ প্রান্তের এগারো নম্বর কাঠিতে এসে পেঁছই, তাহলে দেখা যাবে, এই কাঠিতে কোনও সংখ্যা, ধরা যাক 9, নির্দেশ করলে তার আসল মান হবে,

$$9 \times 10,000,000,000$$

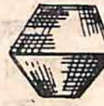
বা ন' হাজার কোটি !

সুতরাং 'সুয়ান পান'-এ সবচেয়ে বড় যে সংখ্যা দেখানো সম্ভব, তার মান, 99,999,999,999 বা ন' হাজার ন'শো নিরেনব্বুই কোটি নিরেনব্বুই লক্ষ নিরেনব্বুই হাজার ন'শো নিরেনব্বুই। তখন এগারোটা কাঠিতেই 9 সংখ্যাটি নির্দেশ করতে হবে।

এ থেকেই বোঝা যাচ্ছে, কত বড় বড় সংখ্যা নিয়ে 'সুয়ান-পান'-এ হিসেব করা সম্ভব। আমাদের দেশে কোনও কোনও চীনে রেস্তরাঁ অথবা লন্ড্রিতে এখনও চোখে পড়ে চীনে দোকানদার 'সুয়ান পান' হাতে নিয়ে দ্রুত হিসেব করে চলেছে।

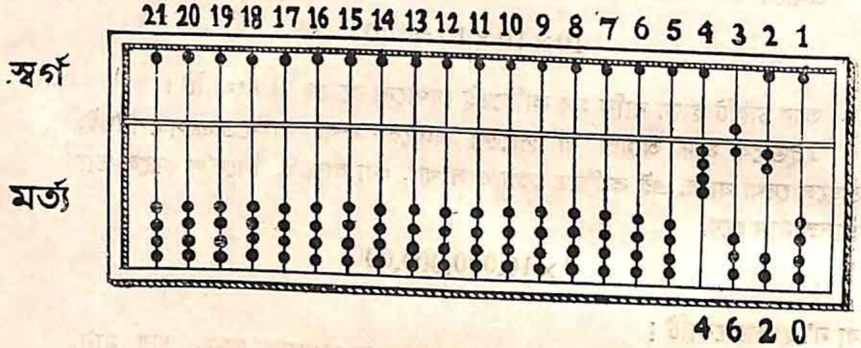
জাপানের অ্যাবাকাস ঃ সোরোবান

জাপানী 'সোরোবান'-এর পুঁতিগুলো 'সুয়ান পান'-এর থেকে সামান্য ভিন্ন চেহারার। দুটো চায়ের কাপ মুখোমুখি এঁটে তাদের হাতল দুটো বাদ দিয়ে দিলে 'সোরোবান'-এর পুঁতির আকৃতির অনেকটা আঁচ পাওয়া যাবে। পুঁতির চেহারা ছাড়া আরও তিনটে বিষয়ে 'সুয়ান পান'-এর সঙ্গে 'সোরোবান'-এর তফাৎ রয়েছে। এক, অ্যাবাকাসের ওপরের অংশে দুটির বদলে মাত্র একটি করে ঘুঁটি। দুই, নিচের অংশে পাঁচটির বদলে চারটি করে ঘুঁটি। আর শেষ তফাৎটি হল লম্বালম্বি বাঁশের কাঠিগুলোর সংখ্যায়। 'সুয়ান পান'-এ ছিল এগারোটি কাঠি। 'সোরোবান'-এ রয়েছে একুশটি। ফলে, সবচেয়ে বড় যে সংখ্যাটি 'সোরোবান'-এ নির্দেশ করা যাবে, সেটা হল একুশটা 9 পাশাপাশি লিখে ফেললে যে সংখ্যা পাওয়া যায়, সেইটি।



সোরোবান-এর পুঁতি

এই অ্যাবাকাসটিও 'সুয়ান পান'-এর মতো দুভাগে ভাগ করা। জাপানীরা ওপরের ভাগটাকে বলে 'স্বর্গ', এবং নিচের ভাগকে বলে 'মর্ত্য'। ছবিতেই দেখা যাচ্ছে, স্বর্গে একটি করে পুঁতি, আর মর্ত্যে চারটি করে। অনেক



সোরোবান

আগে মর্ত্যে পাঁচটা ঘুঁটিই ছিল, কিন্তু 1920 সাল নাগাদ পাঁচ নম্বর ঘুঁটিটার দরকার নেই বুঝতে পেরে সেটিকে নির্বাসন দেওয়া হয়।

প্রতি কাঠিতে ওপরের ভাগে একটি ও নিচের ভাগে চারটে ঘুঁটিই যে যথেষ্ট সেটা খুব সহজেই বোঝা যায়।

প্রত্যেকটি কাঠিতে সবচেয়ে বড় যে সংখ্যাটি দেখানো দরকার সেটি হল 9। স্বর্গের ঘুঁটির মান 5, এবং মর্ত্যের এক একটি ঘুঁটির মান 1। অতএব স্বর্গ ও মর্ত্য মিলিয়ে মোট পাওয়া সংখ্যার মান,

$$5+4=9$$

সুতরাং এই হিসেব থেকেই জাপানীরা প্রতি কাঠিতে দু-দুটো ঘুঁটি কমিয়ে ফেলতে পেরেছিল।

ছবির 'সোরোবান'-এ 4620 সংখ্যাটি দেখানো রয়েছে।

জাপানে 'সোরোবান'-এর চলন এখনও কিছুটা রয়েছে। বিভিন্ন ব্যাঙ্ক ও বড় বড় ব্যবসায়িক প্রতিষ্ঠানে আধুনিক টেবিল-কম্পিউটার চালু হলেও ছোট-খাটো ব্যবসায়ী বা দোকানদারদের কেউ কেউ এখনও 'সোরোবান' ব্যবহার করে। এছাড়া, কিছুদিন আগেও প্রতি বছর জাপানে অ্যাবাকাস প্রতিযোগিতা হত এবং বহু মানুষ তাতে নাম দিত।

টেবিল-কম্পিউটারের সঙ্গে অ্যাবাকাসের প্রতিদ্বন্দ্বিতার অনেক ঘটনাই শোনা গেছে। কোনও চীনা বা জাপানী 'অ্যাবাসিস্ট' বা অ্যাবাকাস-বিশেষজ্ঞকে মুখোমুখি হতে হয়েছে মার্কিনী কম্পিউটার-অপারেটারের সঙ্গে।

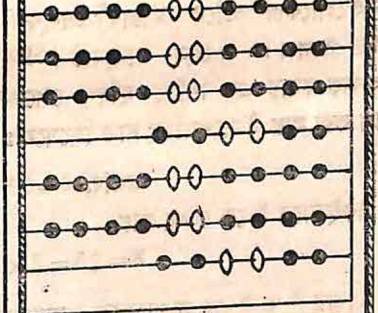
এ ধরনের সবচেয়ে নামী যে-প্রতিযোগিতা হয়েছিল, সেটা অনুষ্ঠিত হয় 1946-এ টোকিওতে। দক্ষতার লড়াইয়ে মুখোমুখি হলেন টমাস উড এবং কিয়োশি মাৎসুজাকি। প্রায় সমস্ত রকমের অঙ্কেই মাৎসুজাকি তাঁর অ্যাবাকাস দিয়ে হারিয়ে দিয়েছিলেন টমাস উডকে। শুধু বিশাল বিশাল সংখ্যার গুণফল বের করার সময় উডের কম্পিউটার অ্যাবাকাসের চেয়ে এগিয়ে ছিল।

অ্যাবাকাসে অঙ্ক কষার গতি এত দ্রুত হওয়ার একটাই মাত্র কারণ ঃ অঙ্কের বেশির ভাগ হিসেব-নিকেশই হয় 'অ্যাবাসিস্ট'-এর মগজে। শুধু অঙ্কের কয়েকটি ধাপ সে নথিবদ্ধ করে রাখে হাতের অ্যাবাকাসে।

অ্যাবাকাসে হিসেব করার অসুবিধে একটা আছে। সেটা হল, অঙ্কের হয়ে-যাওয়া ধাপগুলো তাতে ধরে রাখা যায় না। ফলে অঙ্ক করতে করতে কোথাও যদি একবার ভুল হয়ে যায়, তাহলে সব উত্তরই ভুল বেরোবে। এই সমস্যার সমাধানে জাপানী ব্যবসায়ীরা একটা মজার পথ বের করেছে। তারা একই অঙ্ক করতে দেয় তিনজন 'অ্যাবাসিস্ট'কে—একই সঙ্গে। যদি তিনটে উত্তরই মেলে, তাহলে তারা ধরে নেয় অঙ্ক ঠিক হয়েছে। অর্থাৎ, অনেকটা আমাদের 'তিন সত্যি'র মতো ঃ তিনবার কোনও কথা বললেই সেটা হয়ে যায় নির্ভুল সত্যি।

রাশিয়ার অ্যাবাকাস ঃ স্কোটি

চীন বা জাপানী অ্যাবাকাসের সঙ্গে রুশ দেশের 'স্কোটি'র তফাৎ অনেকটাই। সম্ভবত আরবদের কাছ থেকে রুশরা এই অ্যাবাকাসের ব্যবহার শেখে। এখনও মধ্য প্রাচ্যে এর ব্যবহার আছে। তুর্কি'রা একে বলে 'কুল্‌বা', আর্মেনীয়রা বলে 'ছোরেব্‌'।



স্কোটি

বর্তমান রাশিয়ায় 'স্কোটি'র ব্যবহারের ছবিটা যেন জাপানেরই মতো। কোনও কোনও ছোট খাটো দোকানদার এখনও এই 'যন্ত্রগণক'টি ব্যবহার করে থাকে, যদিও আধুনিক ক্যালকুলেটর ক্রমশ সেখানে জায়গা করে নিচ্ছে।

'স্কোটি'তে অনুভূমিক-ভাবে মোট দশটা সমান্তরাল তার বা রড থাকে। দশটির মধ্যে আটটি রডের

প্রত্যেকটিতে দশটা করে ঘুঁটি থাকে, আর বাকি দুটো রডে চারটে করে। ঘুঁটিগুলোর চেহারা অনেকটা চ্যাপ্টা ডিমের মতো। প্রত্যেক সারির ঠিক মাঝখানের দুটো ঘুঁটির রঙ আলাদা। 'সোরোবান'-এ যেমন আড়াআড়ি একটা লাঠি দিয়ে স্বর্গ ও মর্ত্য, দুটো ভাগ করা থাকে, 'স্কোটি'-তে এই ভাগের দায়িত্ব নেয় ভিন্ন রঙের ঘুঁটি দুটো। রাশিয়ার টাকা রুবল্ ও কোপেক্-এর ভগ্নাংশের হিসেবের জন্য 'স্কোটি'-র দুটো রডে চারটে করে ঘুঁটি রাখা হয়।

সম্প্রতি অন্ধ ছেলেমেয়েদের অঙ্ক শেখানোর কাজে অ্যাবাকাসের গুরুত্ব অনেকে বুঝতে পেরেছে। ফলে নতুন ধরনের কিছু অ্যাবাকাসের চলন শুরু হয়েছে ঃ এই যন্ত্রগুলোয় ঘুঁটি নাড়াচাড়া করতে যে ঘর্ষণের বাধা, সেটাকে আধুনিক প্রক্রিয়ায় অনেক কমিয়ে রড ও ঘুঁটিকে অনেক মসৃণ করে ফেলা হয়েছে।

অ্যাবাকাসে হাতেখড়ি দিতে হলে যোগ অঙ্ক দিয়ে শুরু করতে হয়। তারপর একে একে আসে বিয়োগ, গুণ এবং ভাগ। অ্যাবাকাসে অঙ্ক করতে যারা সিদ্ধ-হস্ত, কোনও হিসেব-নিকেশের সময় তাদের হাত চলা দেখলে মনে হবে, কোনও-রকম ভাবনা-চিন্তা না করেই যন্ত্রের মতো ক্ষিপ্রতায় তারা ঘুঁটিগুলো নিয়ে নাড়াচাড়া করে চলেছে। এই দক্ষতা সত্যিই অবাক করে দেবার মতো।

নেপিয়ারের লাঠি ও লগারিদ্‌ম্ পদ্ধতি

অ্যাবাকাসের পরে এল যন্ত্রগণক তৈরির এক নতুন ধারা। বিভিন্ন ধাতব যন্ত্রাংশ, স্কেল, গিয়ার, পিনিয়ন ইত্যাদি ব্যবহার করে তৈরি হতে লাগল নতুন নতুন গণনা-যন্ত্র। এর সময়টা ছিল মোটামুটিভাবে সপ্তদশ শতকের শুরু।

নতুন বিপ্লবের সূচনা ঘটালেন স্কটল্যাণ্ডের অঙ্কবিদ জন নেপিয়ার। তিনি জ্যোতিষচর্চা করতেন এবং ঘোরতর ঈশ্বর-বিশ্বাসী ছিলেন। কিন্তু তা সত্ত্বেও স্থানীয় লোকেরা তাঁকে মায়াবী জাদুকর বলে সন্দেহ করত। কেউ কেউ বা ভাবত পাগল। এ-হেন মানুষটি 1594 সালে, চুয়াল্লিশ বছর বয়সে, নতুন ভাবে অঙ্ক কষার এক বিচিত্র হদিশ পেলেন। তিনি দেখলেন, 4 সংখ্যাটিকে প্রকাশ করা যায় 2 সংখ্যাটির ঘাত হিসেবে ঃ

$$\text{অর্থাৎ, } 4 = 2^2 = 2 \times 2$$

একইভাবে 8 কে লেখা যায়

$$8 = 2^3 = 2 \times 2 \times 2$$

আবার 4 ও 8-এর গুণফলকে প্রকাশ করা যায় এইভাবে ঃ

$$4 \times 8 = 32$$

$$\text{বা, } 2^2 \times 2^3 = 2 \times 2 \times 2 \times 2 \times 2 = 2^5 = 2^{2+3} = 32$$

সুতরাং 4 ও 8 কে গুণ না করে যদি তাদের 2 দ্বারা প্রকাশিত রাশির ঘাত দুটিকে (2 ও 3) যোগ করা যায় তাহলেই উত্তর মিলে যায় $32 = 2^5$। নেপিয়ার তখন সিদ্ধান্তে পেঁছিলেন ঃ এইভাবে যে কোনও সংখ্যাকে নিশ্চয়ই 2 সংখ্যার কোনও না কোনও ঘাতে প্রকাশ করা সম্ভব। হয়ত সেই ঘাত পূর্ণ সংখ্যা না হয়ে কোনও ভগ্নাংশও হতে পারে। নেপিয়ার দেখলেন, এই অদ্ভুত প্রক্রিয়ায় গুণ ক্রিয়াটি পরিণত হচ্ছে যোগ-এ, এবং ভাগ পরিণত হয় বিয়োগ-এ। তিনি এর নাম দিলেন 'লগারিদ্ম্' পদ্ধতি, যার অর্থ আনুপাতিক সংখ্যা। লগারিদ্ম্ শব্দটি আজও অঙ্কশাস্ত্রে প্রচলিত রয়েছে।

নতুন চিন্তা নিয়ে দীর্ঘ দুই দশক সাধনার পর জন নেপিয়ার 1614 সালে তাঁর লগারিদ্ম্ টেব্ল্ প্রকাশ করেন। এ যুগের কম্পিউটার আমাদের জন-জীবনে যে আলোড়ন ফেলেছে ঠিক একই ধরনের আলোড়ন সৃষ্টি করেছিল নেপিয়ারের লগ-টেব্ল্। এই তালিকার সাহায্যে অঙ্ক কষার ভীষণ সুবিধে হয়ে গেল। বিশেষ করে বিজ্ঞানীরা যেন অপার স্বস্তি পেলেন ঃ ছক বাঁধা ক্লান্তিকর অঙ্ক কষে তাঁদের আর সুদীর্ঘ সময় খরচ করতে হবে না। নেপিয়ারের তালিকা সামনে রেখে তাঁরা নামতার ছকের মতো সুবিধে পেতে লাগলেন। কিছুদিন পরে ইংল্যাণ্ডের আর এক অঙ্কবিদ হেনরী ব্রিগ্স্ নেপিয়ারের তালিকার গুরুত্বপূর্ণ উন্নতিসাধন করেন। বর্তমানে দু-ধরনের লগারিদ্ম্ই প্রচলিত রয়েছে ঃ নেপিয়ারের পদ্ধতিকে সাধারণত বলা হয় 'ন্যাচারাল' লগারিদ্ম্, এবং ব্রিগ্স্-এর রীতিকে বলা হয় 'কমন্' লগারিদ্ম্।

1617 সালে জন নেপিয়ার বিশেষভাবে দাগ কাটা কতকগুলো স্কেল চালু করেন। তিনি এগুলোর নাম দেন 'ক্যালকুলেটিং রড্স্'। তাঁর 'র‍্যাব্-ডোলজিয়া' গ্রন্থে এই রডগুলোর ব্যবহার ব্যাখ্যা করেন নেপিয়ার। এই রডগুলো জনসাধারণের মধ্যে 'নেপিয়ার্স বোন্স্' নামে প্রচলিত ছিল। তাঁর ক্যালকুলেটিং রড্স্ বা বোন্স্ শুধু যে ইওরোপে জনপ্রিয় হয়েছিল তা নয়, চীন এবং জাপানও এই যন্ত্রগণকের দিকে মনোযোগ দিয়েছিল। সামান্য চিন্তা করলেই সিদ্ধান্তে আসা যায়, নেপিয়ারের দণ্ড বা হাড় বয়ে নিয়ে এসেছিল বর্তমানকালের 'স্লাইড রুল'-এর ইঙ্গিত। ডাক্তারের যেমন স্টেথো-স্কোপ, তেমনি ইঞ্জিনীয়ারদের অবিচ্ছেদ্য যন্ত্র হল স্লাইড রুল—যদিও ইলেকট্রনিক ক্যালকুলেটরের দাপটে স্লাইড রুল বর্তমানে অবলুপ্তির পথে।

নেপিয়ারের দণ্ডের মধ্যে 'স্লাইড' বা বিসর্পণ এর কোনও ব্যাপার ছিল না। এই ধর্মটি প্রথম চালু করেন ইংল্যাণ্ডের বিজ্ঞানী উইলিয়াম অট্রেড। লগারিদ্মিক পদ্ধতিতে ঘর কাটা দুটি স্কেল তৈরি করেন তিনি। তারপর একটিকে অপরটির গা ঘেঁষে চলাফেরা করিয়ে লগারিদ্ম্-এর সাহায্যে চটপট গুণ-ভাগ কষে সকলকে তাক লাগিয়ে দেন। 1622 সালে উদ্ভূত এই যন্ত্রটিকে

আদি স্লাইড রুল বলা যেতে পারে। বিজ্ঞানী অট্রেড এর নাম দিয়েছিলেন 'অ্যাস্ট্রোলেব্'—সম্ভবত যন্ত্রটির জ্যোতির্বিদ্যাসংক্রান্ত ব্যবহারের কথা মনে রেখে।

প্রথম মেকানিকাল কম্পিউটার

ঠিক এক বছর পরেই জনৈক অধ্যাপক দৃশ্যপটে হাজির হলেন। নাম উইলিয়াম শিকার্ড—বাইবেলের ভাষা এবং জ্যোতির্বিদ্যায় পারদর্শী। নানান পরিকল্পনা করে অবশেষে একটি যন্ত্রগণক তৈরি করলেন তিনি। যন্ত্রটি যোগ, বিয়োগ, গুণ ও ভাগ করতে পারত। কিন্তু কিছু কিছু ব্যবহারিক অসুবিধে থাকায় যন্ত্রটি জনপ্রিয় হয়নি, এবং পরে এক অগ্নিকাণ্ডে যন্ত্রটি নষ্ট হয়ে গেলে সেটি আর দ্বিতীয়বার তৈরি করা যায়নি।

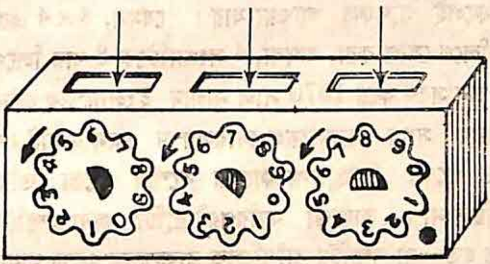
প্রথম সফল 'মেকানিকাল' বা যান্ত্রিক কম্পিউটার তৈরি করতে সমর্থ হন প্রখ্যাত ফরাসী বিজ্ঞানী ব্লাজ পাস্কাল। শৈশব থেকেই তিনি রুগ্ন ছিলেন। কিন্তু তাঁর তীক্ষ্ন মেধার পরিচয়ও একই সঙ্গে পাওয়া গিয়েছিল। পাস্কালের বাবার ইচ্ছে ছিল ছেলেকে প্রাচীন ভাষা শেখাবেন। সুতরাং তাঁকে কোনও অঙ্ক বইয়ের ধারেকাছেও ঘেঁষতে দেননি তিনি। কিন্তু তা সত্ত্বেও পাস্কাল আকৃষ্ট হলেন জ্যামিতির প্রতি। যখন বাবাকে জিজ্ঞেস করে জানলেন, জ্যামিতি হচ্ছে গঠন ও আকার বিষয়ক শাস্ত্র, তখন সম্পূর্ণ একক প্রচেষ্টায় ইউক্লিডের জ্যামিতির প্রথম বত্রিশটি উপপাদ্য ক্রমানুসারে আবিষ্কার করে ফেললেন। পাস্কালের বোন এ-কাহিনী শোনালেও এটি বিশ্বাস হওয়া খুব কঠিন। তবে এ-ঘটনা থেকে পাস্কালের মেধা ও অধ্যাবসায়ের কিছুটা পরিচয় পাওয়া যায়। সুতরাং তিনিই যে 1642 সালে, মাত্র উনিশ বছর বয়েসে, একটি যান্ত্রিক কম্পিউটার তৈরি করে ফেলবেন, এ আর আশ্চর্য কি!

পাস্কালের যন্ত্রের প্রধান অংশ ছিল কয়েকটি দাঁতওয়ালা চাকা—অনেকটা গিয়ার হুইলের মতো। এই চাকাগুলোর মোট দশটি করে দাঁত—প্রতিটি দাঁতে 0 থেকে 9 সংখ্যাগুলো লেখা আছে। মোটর গাড়ির মাইল-মিটারের যে নীতি, অনেকটা সেই নীতিতেই তৈরি হয়েছিল যন্ত্রটি, এবং তার চেহারা ছিল আজকের দিনের মেকানিকাল ডেস্ক ক্যালকুলেটরের মতো।

ছবিতে যে-তিনটি দাঁতওয়ালা চাকা দেখানো হয়েছে তার মধ্যে ডান দিকেরটি সর্বদা এককের ঘরের সংখ্যা নির্দেশ করে (যেমন, 8 সংখ্যাটি), মাঝেরটি দশকের ঘরের সংখ্যা দেখায় (7 সংখ্যাটি), এবং একেবারে বাঁদিকেরটি নির্দেশ করে শতকের ঘরের সংখ্যা (6 সংখ্যাটি)। এখন মনে করা যাক, ছবিতে দেখানো 678 সংখ্যাটির সঙ্গে আমরা 211 যোগ করতে চাই। তাহলে দুটি

সংখ্যার একক-দশক-শতকের ঘরের সংখ্যাগুলোকে পরস্পর যোগ করতে হবে। অর্থাৎ, 6 এর সঙ্গে 2, 7 এর সঙ্গে 1, এবং 8 এর সঙ্গে 1। পাস্কালের যন্ত্রে যোগ করার নিয়ম হল, যে সংখ্যাটি যোগ করতে হবে, দাঁতওয়ালা চাকাটিকে তীর চিহ্নিত দিকে ঠিক ততগুলো দাঁত ঘুরিয়ে দিতে হবে। অর্থাৎ, বাঁদিকের

সংখ্যা দেখার জানালা



পাস্কালের গণনা যন্ত্র

চাকাটিকে ছবির অবস্থা থেকে তীর চিহ্নিত দিকে দু' দাঁত ঘোরাতে হবে, মাঝের চাকাটি ঘুরবে এক দাঁত, এবং শেষেরটিও এক দাঁত। তাহলেই দেখা যাবে, সংখ্যা দেখার জানালায় যথাক্রমে 8, 8 ও 9 সংখ্যাগুলো বা 889 সংখ্যাটি দেখা যাচ্ছে—যা হল 678 ও 211 এর যোগফল।

বিয়োগের ক্ষেত্রেও একই নীতি অনুসরণ করতে হয় পাস্কালের যন্ত্রে। তবে চাকাগুলো তখন ঘোরাতে হয় তীর চিহ্নের বিপরীত দিকে।

পাস্কাল যন্ত্রটি তৈরি করেছিলেন মূলত তাঁর বাবার প্রয়োজনে। কারণ তাঁর বাবা ছিলেন সরকারী কর্মচারী, বিভিন্ন কর নির্ণয়ের জন্য তাঁকে প্রচুর যোগ-বিয়োগ-গুণ-ভাগ করতে হত। যন্ত্রটি তৈরির পর পাস্কাল তার পেটেণ্ট নিলেন, এবং একটি মডেল পাঠিয়ে দিলেন 'রয়াল প্যাট্রন অফ লার্নিং' সুইডেনের রাণী ক্রিশ্চিয়ানার কাছে। রাজ-অনুমতি সাপেক্ষে পাস্কাল পঞ্চাশটিরও বেশি ক্যালকুলেটর তৈরি করেন। তাদের নাম দেওয়া হয় 'পাস্কালাইন'। তিনি চেয়েছিলেন যন্ত্রটির জনপ্রিয়তা ও বাণিজ্যিক ব্যবহার। কিন্তু যন্ত্রটির দাম যথেষ্ট বেশি হওয়ায় পাস্কালের স্বপ্ন সফল হয়নি।

সফল না হলেও পাস্কাল কালে কালে তাঁর যন্ত্রের উন্নতি ঘটানোর চেষ্টা করেছিলেন। 1649 সালে তিনি এই গবেষণার জন্য রাজ-অনুদান পান, এবং প্যারিসের সংগ্রহশালায় তাঁর যে যন্ত্রটি আজও সংরক্ষিত আছে সেটি কিছুটা উন্নত ধরনের। কারণ সেই যন্ত্রটি ছয় অঙ্কের সংখ্যা নিয়ে গণনা করতে পারে। তবে শুধুমাত্র যোগ এবং বিয়োগ, এই দু'রকম গণনা করা যেত পাস্কালের যন্ত্রগণকে।

অন্যান্য মেকানিকাল কম্পিউটার

পাস্কাল নিজে হয়ত ভাবেননি যান্ত্রিক কম্পিউটারের গঠন অন্যরকম করে তিনি পৃথিবীর কম্পিউটারের ইতিহাস বদলে দেবেন। অথচ কাজে তাই হয়েছিল। পাস্কালের যন্ত্রগণকের সাহায্যে যে গুণফল বের করা সম্ভব এটা উপলব্ধি করতে বিজ্ঞানীদের বেশি সময় লাগল না। তাঁরা বললেন, 'বার বার যোগের ফলেই গুণফল পাওয়া যায়। যেমন, 8 × 4 এর অর্থ 8 সংখ্যাটিকে 4 বার লিখে যোগ করা, অথবা, 4 সংখ্যাটিকে 8 বার লিখে যোগ করা।' এই নীতিকে অনুসরণ করে 1670 সাল নাগাদ ইংল্যাণ্ডের জনৈক কূটনীতিবিদ এবং আবিষ্কারক স্যার স্যামুয়েল মোরল্যাণ্ড এমন একটি যন্ত্র তৈরি করলেন যা গুণ করতে পারে। কিন্তু পাস্কালের যন্ত্রের মতো তাঁর যন্ত্রও তেমন নির্ভরযোগ্য ছিল না। তাছাড়া কারিগরী ত্রুটির জন্য যন্ত্রটি খুব সাবধানে ব্যবহার করতে হত—যা যন্ত্রটির বাণিজ্যিক ব্যবহারের পক্ষে বাধা হয়ে দাঁড়ায়।

পাস্কাল ও মোরল্যাণ্ডের যন্ত্রগণকের ত্রুটি-বিচ্যুতি ভালো করে পর্যবেক্ষণের পর 1671 সালে, মাত্র পঁচিশ বছর বয়েসে, জার্মান বিজ্ঞানী গটফ্রিদ উইলহেলম্ লাইপ্‌নিৎস একটি যন্ত্রগণক তৈরি করেন। 'সপ্তদশ শতাব্দীর অ্যারিস্টটল' নামে অভিহিত এই দার্শনিক তথা অঙ্কবিদের যন্ত্রটি যোগ-বিয়োগ গুণ-ভাগ প্রক্রিয়া ছাড়াও বিভিন্ন সংখ্যার বর্গমূল নির্ণয় করতে পারত। যেমন, 25 এর বর্গমূল 5, 81 এর বর্গমূল 9। 1694 সালে যন্ত্রটির একটি কার্যকরী মডেল তৈরি হল। লাইপ্‌নিৎস্ তার নাম দিলেন 'রেকনিং মেশিন'। যন্ত্রটি প্রদর্শিত হল লণ্ডনের রয়াল সোসাইটিতে। কিন্তু এ ক্ষেত্রেও যান্ত্রিক ত্রুটি মেশিনটির জনপ্রিয়তার পথে বাধা হয়ে দাঁড়ায়।

লাইপ্‌নিৎস্-এর যন্ত্রেই প্রথম সফলভাবে পৌনঃপুনিক যোগের সাহায্যে গুণফল নির্ণয় করা হয়েছিল। ঘটনাটি উল্লেখযোগ্য, কারণ আধুনিক কম্পিউটারও একই পদ্ধতিতে গুণফল নির্ণয় করে থাকে।

এরপর যন্ত্রগণকের বিবর্তন-পথে শুরু হয় দ্রুত পালাবদল। একের পর এক উন্নত মানের গণক তৈরির প্রচেষ্টা চালান ইতালির পাদুয়া বিশ্ববিদ্যালয়ের জ্যোতির্বিদ্যার অধ্যাপক মার্শিজ গিওভানি পোলেনি (1709 সাল), জার্মানীর লিপজিগ শহরের এক মেকানিক জ্যাকব লিউপোল্ড—মৃত্যুর ঠিক আগে তিনি একটি যন্ত্র তৈরি করেন (1727 সাল), এবং গিসেন-এর অঙ্কশাস্ত্রের অধ্যাপক ক্রিশ্চিয়ান লুডউইগ গারস্টেন—তিনি স্বআবিষ্কৃত একটি যন্ত্রের বর্ণনা লিপিবদ্ধ করে যান 1735 সালে।

কিন্তু প্রথম যে যন্ত্রগণক বাণিজ্যিক সফলতা লাভ করে সেটি তৈরি করেন ফ্রান্সের আবিষ্কারক চার্লস জেভিয়ার টমাস। 1820 সালে নির্মিত এই যন্ত্র-

গণকটির নাম ছিল 'অ্যারিথোমিটার'। আগের যন্ত্রগুলোর তুলনায় উন্নত এই গণকটিতে যোগ, বিয়োগ, গুণ এবং ভাগ—চার রকম অঙ্কই কষা যেত। পরবর্তী ষাট বছরে গুণফল নির্ণয়ের প্রায় দেড় হাজারটি যন্ত্র তৈরি করেছিলেন টমাস। আশ্চর্যের কথা, বর্তমান শতাব্দীর ষাটের দশকেও প্যারিসে সে-যন্ত্র তৈরি হত। টমাসের যন্ত্রে টাইপরাইটার মেশিনের মতো বোতাম টিপে বিভিন্ন তথ্য বা সংখ্যা ঢুকিয়ে দেওয়া যেত যন্ত্রগণকের মধ্যে, এবং পরবর্তী গণনার সময় সেগুলো প্রয়োজন মতো ব্যবহার করা হত। এছাড়া কোনও বিশাল অঙ্কের বিভিন্ন ধাপের ফলাফল সঞ্চিত করে রাখা যেত যন্ত্রের একটি বিশেষ অংশে, যাকে স্মৃতিভাণ্ডার বা মেমোরি বলা যায়।

চার্লস ব্যাবেজ ও তাঁর 'ইঞ্জিন'

'যান্ত্রিক কম্পিউটার' নাটকের প্রধান অভিনেতা রঙ্গমঞ্চে হাজির হলেন জেভিয়ার টমাসের ঠিক পরেই। তিনি ছিলেন কম্পিউটারশাস্ত্রের 'আইজ্যাক নিউটন', আজকের ডিজিটাল কম্পিউটারের জনক, চার্লস ব্যাবেজ। ব্যাবেজ পেশায় ছিলেন কেম্ব্রিজ বিশ্ববিদ্যালয়ের অধ্যাপক। তিনি আবিষ্কার করেছিলেন ইতিহাস-বিখ্যাত দুটি যন্ত্রগণক, 'ডিফারেন্স ইঞ্জিন' এবং 'অ্যানালিটিক্যাল ইঞ্জিন'। কিন্তু দুটির একটিকেও তিনি সম্পূর্ণ রূপ দিয়ে যেতে পারেননি। অথচ আজ আমাদের চারপাশে শিক্ষা ও প্রযুক্তির বিভিন্ন দিকে ছড়িয়ে রয়েছে ব্যাবেজের তত্ত্বের অজস্র বাস্তব রূপায়ণ—ইলেকট্রনিক ক্যালকুলেটর ও কম্পিউটার। এটাই চার্লস ব্যাবেজের চরম সাফল্য।

চরিত্রে চার্লস ব্যাবেজ মানুষটি ছিলেন ভারী অদ্ভুত। জন্মেছিলেন 1792 সালে ইংল্যাণ্ডের ডেভনশায়ারে এক অবস্থাপন্ন পরিবারে। কিন্তু তাঁর খেয়াল ছিল বিচিত্র রকম, আর আবিষ্কারও করেছিলেন বহু রকমের।

ব্যাবেজ এক ধরনের ডুবোজাহাজ আবিষ্কার করেছিলেন, বলেছিলেন রকেটের নতুন ধরনের ব্যবহারের কথা, বাতিঘরের সংখ্যা কমানোর এক নতুন সাংখ্যিক পদ্ধতি তৈরি করেছিলেন। এছাড়া তাঁর আগ্রহ ছিল ভূতত্ত্ব, প্রত্নতত্ত্ব এবং জ্যোতির্বিদ্যা বিষয়ে। আবার একই সঙ্গে আগ্নেয়গিরি ভিসুভিয়াসের অভ্যন্তরে অভিযানও চালিয়েছিলেন তিনি।

ভাবতে অবাক লাগে, এ-হেন মানুষটি কবিতা লিখতেন, পার্লামেন্টের নির্বাচনে দাঁড়িয়েছিলেন, জীবনবীমা নিয়ে মাথা ঘামিয়েছেন, এমন কি স্পীডো-মিটার এবং সবখোল্ চাবিও আবিষ্কার করেছেন।

ব্যাবেজের মস্তিষ্কে নতুন যন্ত্রগণক তৈরির প্রথম বীজ রোপিত হয় 1812 সালে। নেপিয়ারের লগারিদম্-টেব্‌ল্-এ বেশ কিছু ত্রুটি আবিষ্কার করেন তিনি। এছাড়া জ্যোতির্বিদ্যা বিষয়ক গণনা-তালিকায় একই জাতীয় কয়েকটি

মারাত্মক ভুলের মুখোমুখি হয়ে ব্যাবেজ মন্তব্য করেন, 'মানুষের ওপর বিশ্বাস রাখা কঠিন।' সেই হতাশাপীড়িত সময়ে কিভাবে তাঁর মগজে যান্ত্রিক গণনার সম্ভাবনা প্রথম জন্ম নেয়, সে-বিষয়ে নিজের লেখা 'লাইফ অফ এ ফিলসফার' বইতে ব্যাবেজ একটি ঘটনা উল্লেখ করেছেন ঃ 'একদিন সন্ধ্যাবেলা কেম্ব্রিজ বিশ্ববিদ্যালয়ের অ্যানালিটিক্যাল সোসাইটির ঘরে বসে ছিলাম। সামনের টেবিলে খোলা অবস্থায় রাখা ছিল একটা লগারিদম্-তালিকা। একটু তন্দ্রা মতো এসেছে। এমন সময় সোসাইটির আরেকজন সদস্য এসে ঘরে ঢুকল। আমাকে তন্দ্রাচ্ছন্ন অবস্থায় দেখে ডেকে উঠল, "আরে, ব্যাবেজ, কিসের স্বপ্ন দেখছেন?" সে-কথায় আমি উত্তর দিলাম, "ভাবছি, এসব টেব্ল্গুলো (লগারিদম্-টেব্লের দিকে আঙুল দেখিয়ে) যন্ত্র দিয়ে অঙ্ক কষে তৈরি করলে বেশ হত।"'

এ-থেকেই শুরু হল তাঁর সৃজনশীল ভাবনা।

1822 সাল নাগাদ ব্যাবেজের তাত্ত্বিক গণনা শেষ হল। সম্পূর্ণ হল তাঁর যন্ত্রগণকের নকশা। তিনি যন্ত্রটির নাম দিলেন 'ডিফারেন্স ইঞ্জিন'। এই যন্ত্রটির সাহায্যে যে কোনও সংখ্যার যে কোনও ঘাত নির্ণয় করা যেত ঃ

যেমন, $5^2=25$, $3^4=81$, $2^5=32$ ইত্যাদি।

শুধু তাই নয়, এই গণনা নির্ভুল পাওয়া যেত দশমিক চিহ্নের পর আটটি অঙ্ক পর্যন্ত। এছাড়া নির্ণয় করা যেত যে কোনও ফাংশন বা অপেক্ষকের মান।

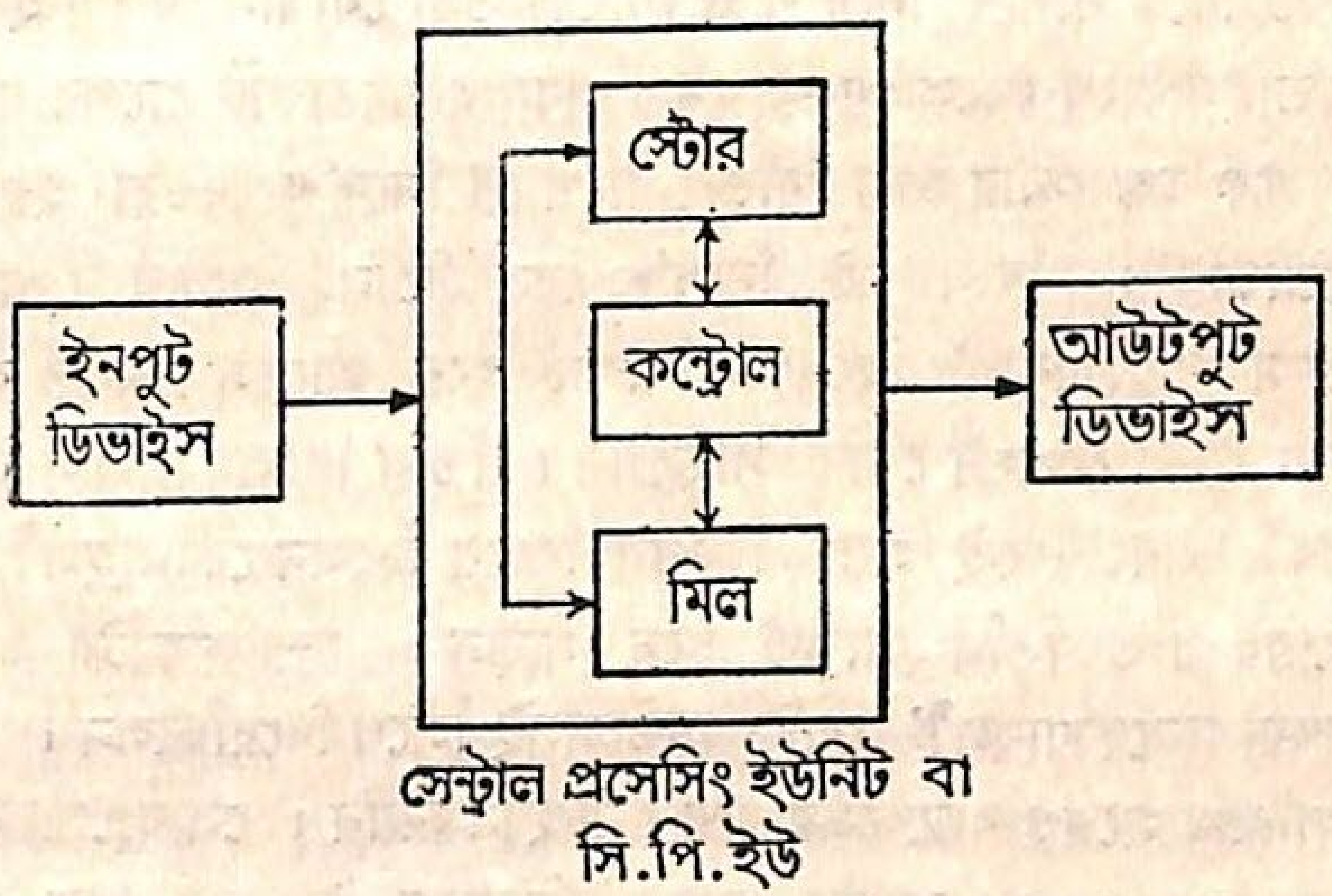
এইসব গুণ সম্পন্ন প্রাথমিক একটি যন্ত্রগণক তৈরি করেছিলেন ব্যাবেজ। কিন্তু সেটিকে উন্নত রূপ দেবার পরিকল্পনায় তিনি রয়াল সোসাইটির মাধ্যমে ব্রিটিশ সরকারের কাছে আর্থিক সাহায্যের আবেদন রাখেন। ব্যাবেজ বলেন, নতুন যন্ত্রটি বিশেষ ব্যবস্থার সাহায্যে নিজের গণনার ত্রুটি নিজেই নির্ণয় করতে পারবে এবং তা সংশোধনও করবে। তাছাড়া কোনও অঙ্কের ফলাফল সরাসরি মুদ্রিত করবে কাগজে।

আপাতভাবে আশ্চর্য ঠেকলেও ঊনবিংশ শতাব্দীর ব্রিটিশ সরকার এ-জাতীয় একটি যন্ত্রের গুরুত্ব উপলব্ধি করেছিলেন। ফলে ব্যাবেজের প্রস্তাবে তাঁরা রাজী হন। কিন্তু দুর্ভাগ্যের কথা, দীর্ঘ বিশ বছরেও যন্ত্রটি সম্পূর্ণ না হওয়ায় 1842-43 সালে ব্রিটিশ সরকার প্রকল্পটি বাতিল করেন। প্রকল্পটির পেছনে সরকারী খরচ হয়েছিল সতেরো হাজার পাউন্ড স্টার্লিং, এবং অবস্থাপন্ন চার্লস ব্যাবেজ নিজেও সম্ভবত একই অঙ্কের অর্থব্যয় করেছিলেন। কিন্তু এই ঘটনার দশ বছর আগেই তাঁর মগজে জন্ম নিয়েছে নতুন ও উন্নততর আর একটি যন্ত্রের বীজ ঃ অ্যানালিটিক্যাল ইঞ্জিন।

1833 সালেই প্রথম যন্ত্রটির ওপর গবেষণা ইত্যাদি বন্ধ করে দেন চার্লস

ব্যাবেজ। তাঁর মাথায় তখন ঢুকেছে নতুন একটি যন্ত্রগণকের পরিকল্পনা। নানারকম গণনায় সক্ষম এই যন্ত্রটির তিনি নাম রেখেছিলেন 'অ্যানালিটিক্যাল ইঞ্জিন'। যন্ত্রটির গঠন ও চরিত্রের প্রধান বৈশিষ্টগুলো বিচার করলে আজকের ডিজিটাল কম্পিউটারের সঙ্গে অ্যানালিটিক্যাল ইঞ্জিনের অনেকটা মিল ছিল। যন্ত্রগণককে নির্দেশ দেবার জন্য ব্যবহার করা হয়েছিল 'পাঞ্চ্‌ড্‌' কার্ড বা ছিদ্রযুক্ত কার্ড। এছাড়া, কোনও অঙ্ক কষার সময়ে তার পথ চলতি ফলাফল-গুলো আবার ব্যবহারের সুবিধের জন্য গণকের ভেতরে সঞ্চিত রাখার ব্যবস্থা ছিল। আর সবচেয়ে অবাক করে দেবার মতো বৈশিষ্ট ছিল, গণকের হিসেবের ফলাফল স্বয়ংক্রিয়ভাবে ছাপা হয়ে যাওয়ার আয়োজন।

এতে স্পষ্টই বোঝা যায়, আজকের আধুনিক কম্পিউটার যে যে রীতিনীতি মেনে কাজ করে ঠিক সেই নীতিগুলোকেই আশ্চর্যভাবে কাজে লাগিয়েছিলেন প্রতিভাধর ব্যাবেজ। তাঁর কম্পিউটারে বিভিন্ন নির্দেশ অনুযায়ী বিভিন্ন কাজ সম্পন্ন করত নানান ধাতব যন্ত্রাংশ—আজকের কম্পিউটারে যে কাজগুলো করে ইলেকট্রনিক বর্তনী।



ব্যাবেজের পরিকল্পিত কম্পিউটারের প্রধান অংশ। বিভিন্ন অংশের মধ্যে তথ্য ও নির্দেশের আদান-প্রদানের অভিমুখ বোঝাচ্ছে তীর চিহ্নের সংকেত।

ব্যাবেজের পরিকল্পনায় অ্যানালিটিক্যাল ইঞ্জিনের মোট ভাগ ছিল তিনটে ঃ স্টোর, মিল এবং কন্ট্রোল। 'স্টোর' বা ভাণ্ডার-এর কাজ হল অঙ্ক কষার সময়ে বিভিন্ন ফলাফল সঞ্চয় করে রাখা—পঞ্চাশ অঙ্কের একহাজারটি সংখ্যা পর্যন্ত সঞ্চয় করার ক্ষমতা ছিল এই যন্ত্রগণকের। বর্তমান যুগে 'স্টোর'কে বলা হয় 'মেমোরি স্টোরেজ' বা স্মৃতিভাণ্ডার।

যন্ত্রের 'মিল' অংশটি ছিল প্রকৃত গণক বা 'ক্যালকুলেটর'। ব্যাবেজ এই

অংশে পাস্কালের মতোই দাঁতওয়ালা চাকার ব্যবহার করেন, তবে গণনার গতি বাড়ানোর জন্য তিনি বিশেষ কারিগরী ব্যবস্থা নেন। তার ফলে যন্ত্রটি প্রতি সেকেণ্ডে একটি করে যোগ করতে পারত, এবং দুটি সংখ্যার গুণফল বের করতে সময় নিত মাত্র এক মিনিট। তবে সব ক্ষেত্রেই সংখ্যাগুলো হল পঞ্চাশ অঙ্কের চেয়ে ছোট সংখ্যা। প্রসঙ্গত বলা যেতে পারে, আধুনিক কম্পিউটার প্রতি সেকেণ্ডে দশ লক্ষেরও বেশি যোগফল নির্ণয় করতে পারে।

ব্যাবেজের যন্ত্রের শেষ প্রধান অংশ 'কণ্ট্রোল'-এর কাজ প্রকৃতপক্ষে দুটি। এক ঃ লক্ষ্য রাখা যে যন্ত্রগণক নির্দিষ্ট ক্রমানুসারে অঙ্কের বিভিন্ন ধাপের নির্দেশ পালন করছে কি না। দুই ঃ অসংখ্য গিয়ার ও চাকার মাধ্যমে 'মিল' এবং 'স্টোর'-এর মধ্যে তথ্যের আদান-প্রদান করা।

বর্তমানকালে উপরোক্ত প্রধান তিন অংশের সম্মিলিত নাম 'সেণ্ট্রাল প্রসেসিং ইউনিট' বা সংক্ষেপে সি. পি. ইউ.।

স্টোর, মিল এবং কণ্ট্রোল ছাড়া আরও দুটি অংশ ছিল ব্যাবেজের যন্ত্রে। তাদের নাম যথাক্রমে ইনপুট ও আউটপুট ডিভাইস। ইনপুট ডিভাইস অংশের মাধ্যমে কম্পিউটারকে নানান নির্দেশ ও বিভিন্ন তথ্য যোগান দেওয়া হত। আর আউটপুট ডিভাইস অংশ থেকে পাওয়া যেত গণনার প্রত্যেকটি ধাপের ফলাফল।

কম্পিউটারকে অঙ্ক কষার জন্য বিভিন্ন ধাপে যে নির্দেশ দেওয়া হয় তাকেই বলা হয় 'প্রোগ্রাম'। যিনি এই নির্দেশ দেন তিনিই হলেন 'প্রোগ্রামার'। পৃথিবীর প্রথম 'প্রোগ্রামার' হিসেবে বিশিষ্ট হয়ে আছেন প্রখ্যাত কবি লর্ড বায়রনের কন্যা আডা অগাস্টা লেডি লাভলেস। তিনি ছিলেন চার্লস ব্যাবেজের ঘনিষ্ঠ বান্ধবী, এবং একই সঙ্গে বিজ্ঞান বিষয়ে ছিলেন কৌতূহলী। ফলে ব্যাবেজের যন্ত্রের প্রতি তিনি আকৃষ্ট হয়ে পড়েন। যন্ত্রগণকটির জন্য বহু 'প্রোগ্রাম' অথবা 'নির্দেশাবলী' লেডি লাভলেসই লিখে দিয়েছিলেন।

এ-হেন পরিশ্রম সত্ত্বেও ব্যাবেজের যন্ত্র ত্রুটিহীন হয়নি। সে-যুগের কারিগরী দক্ষতা তেমন নিপুণ না হওয়ায় অ্যানালিটিক্যাল ইঞ্জিনের বহু যন্ত্রাংশই নিখুঁতভাবে তৈরি করা যায়নি। তবে ব্যাবেজের রেখে যাওয়া নকশা ও কাগজপত্র পরীক্ষা করে স্পষ্টই প্রমাণ পাওয়া গেছে যে তাঁর তত্ত্বগত ধ্যানধারণায় কোনও ভুলত্রুটি ছিল না।

ব্যাবেজের সময়ে যন্ত্রগণক নিয়ে নতুন চিন্তাভাবনা করেছেন আরও বেশ কিছু শিক্ষাবিদ ও প্রযুক্তিবিদ। তবে তাঁরা মূলত প্রচলিত যন্ত্রগুলোর উন্নতি করার চেষ্টাতেই সময় ও মেধা খরচ করেছেন। যেমন 1850 সালে ডর্ ফেল্ট একটি চাবি-চালিত গণকের পেটেণ্ট নেন। তবে যন্ত্রটি শুধুই যোগ ও বিয়োগ করতে পারত। 1886-87 সালে যন্ত্রটিকে আরও উন্নত রূপ দেওয়া হয়।

প্রায় একই সময়ে উইলিয়াম সেওয়ার্ড বারোজ প্রথম সফল বাণিজ্যিক যন্ত্রগণক

তৈরি করেন। সেটিকে পরবর্তীকালে বহুভাবে উন্নত করা হয়। তার মধ্যে উল্লেখযোগ্য সংযোজনটি ঘটে 1889 সালে। যন্ত্রটি গণনার ফলাফল মুদ্রিত করতে শেখে। এরপর মনরো ও মার্চ্যাণ্ট-এর যন্ত্রগণক প্রচলিত হয় 1911 সালে, এবং যন্ত্রগণকে ইলেকট্রিক মোটরের ব্যবহার প্রবর্তন করা হয় 1920 সাল নাগাদ।

ধাতব যন্ত্রাংশ দিয়ে তৈরি কম্পিউটারের যুগ শেষ হয় 1937 সালে, যখন হার্ভার্ড বিশ্ববিদ্যালয়ের হাওয়ার্ড আইকেন আই. বি. এম. কর্পোরেশনের সঙ্গে যুগ্মভাবে গবেষণা করে দীর্ঘ সাত বছরের চেষ্টায় তৈরি করেন পৃথিবীর প্রথম ডিজিটাল কম্পিউটার 'হার্ভার্ড মার্ক-ওয়ান'। তখনই আমরা পেলাম চার্লস ব্যাবেজের পরিকল্পনার সার্থক রূপ।

ইলেকট্রনিক ডিজিটাল কম্পিউটার

চার্লস ব্যাবেজের স্বপ্নকে প্রথম বাস্তব রূপ দিলেন হাওয়ার্ড হ্যাথাওয়ে আইকেন। হার্ভার্ড বিশ্ববিদ্যালয়ের অধ্যাপক আইকেন 1944 সালের মে মাসে তৈরি করলেন পৃথিবীর প্রথম ডিজিটাল কম্পিউটার। এই কাজে তাঁকে সহযোগিতা করেছিল ইণ্টারন্যাশনাল বিজনেস মেশিনস্ কর্পোরেশন বা সংক্ষেপে আই. বি. এম.। 'হার্ভার্ড মার্ক-ওয়ান' নামে এই কম্পিউটারটি ইতিহাসে বিখ্যাত হয়ে আছে। যন্ত্রটি লম্বায় ছিল 51 ফুট, আর উচ্চতায় 8 ফুট। এর ভেতরে প্রায় সাড়ে সাত লক্ষ যন্ত্রাংশ ছিল, আর বিদ্যুৎবাহী তার লেগেছিল 500 মাইলেরও বেশি। সবচেয়ে বড় যে সংখ্যা নিয়ে 'মার্ক-ওয়ান' কাজ করতে পারত তা হল 23 অঙ্কের সংখ্যা। যে কোনও দুটি সংখ্যা যোগ করতে তার সময় লাগতো 0·3 সেকেণ্ড, গুণ করতে সময় নিত প্রায় 6 সেকেণ্ড, আর ভাগ করতে মোটামুটিভাবে 12 সেকেণ্ড।

প্রথম ডিজিটাল কম্পিউটার হলেও 'মার্ক-ওয়ান' পুরোপুরি ইলেকট্রনিক কম্পিউটার ছিল না। এতে বৈদ্যুতিক, ইলেকট্রনিক ও ধাতব যন্ত্রাংশ ব্যবহার করা হয়েছিল। আর সুইচ এবং পাঞ্চড্ কার্ডের সাহায্যে যন্ত্রটিকে তথ্য ও নির্দেশ দেওয়া হত।

'মার্ক-ওয়ান' কাজ শুরু করার সময়েই বিজ্ঞানীরা আঁচ করেছিলেন, ইলেকট্রনিক ডিজিটাল কম্পিউটার আর বেশি দূরে নেই। সুতরাং 1945 সালের ডিসেম্বর মাসে যখন পুরোপুরি ইলেকট্রনিক ডিজিটাল কম্পিউটার 'ইনিয়াক' (ENIAC) চালু হল তখন কেউই বোধহয় তেমন অবাক হননি। 'ইনিয়াক'-এর পুরো নাম ইলেকট্রনিক নিউমারিক্যাল ইন্টিগ্রেটর অ্যাণ্ড কম্পিউটার। পেন্সিলভ্যানিয়া বিশ্ববিদ্যালয়ের 'মুর স্কুল অফ ইলেকট্রি-ক্যাল ইঞ্জিনিয়ারিং'-এর দুই অধ্যাপক জন প্রেস্পার একার্ট জুনিয়র ও জন

উইলিয়াম মশ্‌লি এই কম্পিউটারটি তৈরি করেন। 'মার্ক-ওয়ান'-এর তুলনায় 'ইনিয়াক' কতটা উন্নত ছিল তা একটা উদাহরণ দিলেই বোঝা যাবে। এক ঘণ্টায় 'ইনিয়াক' যে গণনা করতে পারত সেই কাজে 'মার্ক-ওয়ান' সময় নিত প্রায় এক সপ্তাহ।

'ইনিয়াক' মাপে নেহাৎ ছোটখাটো ছিল না। তার ওজন ছিল 30 টন। 100 ফুট লম্বা, 10 ফুট উঁচু আর 3 ফুট পুরু এই যন্ত্রটিতে দুটি সংখ্যা যোগ করতে সময় লাগত 200 মাইক্রোসেকেণ্ড (1 মাইক্রোসেকেণ্ড = 1 সেকেণ্ডের 10 লক্ষ ভাগের এক ভাগ), আর গুণ করতে লাগত 2,800 মাইক্রোসেকেণ্ড।

একার্ট ও মশ্‌লি যে-কাজের গোড়াপত্তন করলেন তা অন্যান্য বিশেষজ্ঞদের চেষ্টায় এগিয়ে চলল অত্যন্ত দ্রুত গতিতে। 'ইনিয়াক'-এর পর একে একে এল 'এডভ্যাক', 'এডস্যাক', 'ইউনিভ্যাক-ওয়ান', 'লিও' ইত্যাদি। আর বিজ্ঞানের ইলেকট্রনিক্স শাখার অগ্রগতির সঙ্গে সঙ্গে ইলেকট্রনিক কম্পিউটার মাপেও ছোট হতে শুরু করল। উদাহরণ হিসেবে মার্কিন ফেয়ারচাইল্ড কোম্পানির 'এফ-এইট' কম্পিউটারের উল্লেখ করা যেতে পারে। সত্তরের দশকের এই কম্পিউটারটি আয়তনে 'ইনিয়াক'-এর 300,000 ভাগের এক ভাগ, আর ওজন মাত্র 1 পাউণ্ড (500 গ্রামেরও কম)। 'ইনিয়াক' চালাতে যতটা বিদ্যুৎ-শক্তি খরচ হত, 'এফ-এইট' নিত তার 56,000 ভাগের এক ভাগ।

কম্পিউটার আবিষ্কার নিয়ে বিতর্ক

ইলেকট্রনিক ডিজিটাল কম্পিউটারের আবিষ্কারক হিসেবে এতদিন একার্ট ও মশ্‌লিকেই বিজ্ঞানী-জগৎ মেনে নিয়েছিলেন, কিন্তু সম্প্রতি মার্কিন আদালত অন্য দুজন বিজ্ঞানীকে এই স্বীকৃতি দিয়েছে। আদালতের রায় অনুযায়ী পদার্থবিদ জন ভিনসেণ্ট অ্যাটানাসফ ও ক্লিফোর্ড বেরি প্রথম ইলেকট্রনিক ডিজিটাল কম্পিউটারের পেটেণ্ট নিয়েছেন 1942 সালে। অ্যাটানাসফের বক্তব্য অনুযায়ী, 1940 সালের গোড়াতেই তিনি সহকারী বেরির সাহায্যে ইলেকট্রনিক কম্পিউটারের একটি নমুনা তৈরি করে ফেলেছিলেন। 1941 সালে ফিলাডেলফিয়ার আর্সিনাস কলেজে ছিলেন মশ্‌লি। তিনি অ্যাটানাসফের ইলেকট্রনিক যন্ত্রের কথা শুনেছিলেন। অ্যাটানাসফের সঙ্গে দেখা করতে তিনি আইওয়া স্টেট কলেজে যান। সেখানে এক সপ্তাহ ধরে দুই বিজ্ঞানীর মধ্যে নানান আলোচনা হয়। এর ঠিক পরেই মশ্‌লি পেন্সিলভ্যানিয়া বিশ্ববিদ্যালয়ের 'মুর স্কুল অফ ইলেকট্রিক্যাল ইঞ্জিনিয়ারিং'-এ যোগ দেন। তারপর যথাসময়ে জন্ম নেয় 'ইনিয়াক'।

আদালত অ্যাটানাসফের পক্ষে রায় দিলেও বিজ্ঞানী-মহলে এই বিতর্কের এখনও পুরোপুরি নিষ্পত্তি হয়নি।

কম্পিউটার জেনারেশন বা প্রজন্ম

প্রথম যে ইলেকট্রনিক কম্পিউটার তৈরি হয়েছিল তাতে ইলেকট্রনিক যন্ত্রাংশ হিসেবে ব্যবহার করা হয়েছিল ডায়োড ভাল্ব, ট্রায়োড ভাল্ব ইত্যাদি। এই ভাল্বগুলো অনেকটা কাচের বাল্বের মতো দেখতে। পুরোনো আমলের বড়সড় রেডিওতে এইরকম ইলেকট্রনিক ভাল্ব ব্যবহার করা হত। ভাল্ব-গুলো মাপে যেমন বড়, তেমনি এদের পিছনে বিদ্যুৎ খরচও কম নয়। 'ইনিয়াক' এর বিদ্যুৎ খরচের হার ছিল প্রতি ঘণ্টায় 1,40,000 ইউনিট (1 ইউনিট = 1 কিলোওয়াট-ঘণ্টা)।

ইলেকট্রনিক ভাল্ব ব্যবহার করে যে-সব কম্পিউটার তৈরি হয়েছিল তাদের বলা হয় প্রথম জেনারেশন বা প্রথম প্রজন্মের কম্পিউটার। যেমন, 'ইনিয়াক', 'এডভ্যাক', 'ইউনিভ্যাক-ওয়ান' ইত্যাদি হল প্রথম প্রজন্মের কম্পিউটার।

এরপর 1948 সালে আবিষ্কার হল ট্রানজিস্টার। সেমিকণ্ডাক্টর বা অর্ধপরিবাহী পদার্থ দিয়ে তৈরি এই ইলেকট্রনিক যন্ত্রাংশটি ভাল্বের তুলনায় মাপে অনেক ছোট আর বিদ্যুৎও খরচ করে অনেক কম। সুতরাং খুব সহজেই প্রথম প্রজন্মের কম্পিউটারের ইলেকট্রনিক ভাল্বগুলোর বদলে বিজ্ঞানীরা ট্রানজিস্টারের ব্যবহার শুরু করলেন। আর এইভাবেই এল দ্বিতীয় প্রজন্মের যুগ। ট্রানজিস্টার ব্যবহার করার ফলে দ্বিতীয় প্রজন্মের কম্পিউটার প্রথম প্রজন্মের চেয়ে মাপে বেশ ছোট হয়ে গেল।

ট্রানজিস্টার আবিষ্কারের পরে ইলেকট্রনিক্স-বিজ্ঞানের উন্নতি হতে লাগল অত্যন্ত দ্রুত গতিতে। ফলে তার সঙ্গে তাল মিলিয়ে কম্পিউটারও উন্নত হতে শুরু করল। তৃতীয় প্রজন্মের কম্পিউটারে ব্যবহার করা হল ইণ্টিগ্রেটেড সার্কিট বা আই. সি। ইণ্টিগ্রেটেড সার্কিটে অনেক জটিল বর্তনীকে সূক্ষ্ম মাপে খুব ছোট জায়গার মধ্যে তৈরি করে ফেলা হয়। এই জন্যেই এর নাম ইণ্টিগ্রেটেড সার্কিট বা আই. সি. চিপ। আই. সি. ব্যবহারের ফলে কম্পিউটার মাপে আরও ছোট হল। তখন কম্পিউটারের ক্ষমতা বাড়াতে গিয়ে তার মাপ আর কখনও বাধা হয়ে দাঁড়াল না।

আই. সি. আবিষ্কার করেও বিজ্ঞানীরা কিন্তু থেমে থাকলেন না। ইলেক-ট্রনিক বর্তনীকে সূক্ষ্ম থেকে সূক্ষ্মতর করে তোলার জন্য তাঁদের গবেষণা ও নিরলস পরিশ্রম চলতেই থাকল। এইভাবেই আবিষ্কার হল ভি. এল. এস. আই. পদ্ধতি—যার পুরো নাম হল ভেরি লার্জ স্কেল ইণ্টিগ্রেশান। এই পদ্ধতিতে লক্ষ লক্ষ ইলেকট্রনিক যন্ত্রাংশকে খুব ছোট্ট জায়গার মধ্যে ঢুকিয়ে

দেওয়া গেল। ফলে কম্পিউটার মাপে ছোট হতে হতে একটা বইয়ের চেহারা নিল। বাজারে প্রচলিত হল টেব্‌ল্‌-কম্পিউটার, পকেট-কম্পিউটার, আর এ ছাড়া বেশি ক্ষমতাসম্পন্ন বড় মাপের কম্পিউটার তো আছেই। এই প্রজন্মের কম্পিউটারগুলোকে বলা হয় চতুর্থ প্রজন্মের কম্পিউটার।

বিভিন্ন কম্পিউটার প্রজন্মের শ্রেণীবিভাগ নিয়ে বিজ্ঞানীরা সকলেই যে একমত তা নয়। কেউ কেউ কম্পিউটারের গঠন-পদ্ধতির ওপরে ভিত্তি করে শ্রেণীবিভাগ করেছেন। আবার অন্যেরা তার সঙ্গে সঙ্গে কম্পিউটারের কার্য-পদ্ধতিও বিবেচনা করেছেন। তবুও মোটামুটিভাবে এ-পর্যন্ত কম্পিউটারকে চারটি প্রজন্মে ভাগ করা গেছে।

ইলেকট্রনিক্স-বিজ্ঞান এখনও যে গতিতে এগিয়ে চলেছে তাতে পঞ্চম প্রজন্মের কম্পিউটার যে আরও নতুন চমক নিয়ে আসবে তাতে কোনও সন্দেহ নেই। জাপান প্রথম ঘোষণা করেছে পঞ্চম প্রজন্মের কম্পিটারের কথা। প্রযুক্তি ও কারিগরী পদ্ধতির দিক থেকে এই প্রজন্মের কম্পিউটার আগের চারটি প্রজন্মের থেকে অনেকটাই আলাদা। জাপান যে নতুন প্রযুক্তির কথা জানিয়েছে তাতে একটিমাত্র চিপের ওপরে এক কোটি ট্রানজিস্টার জায়গা করে নেবে। আর এই প্রজন্মের কম্পিউটার কাজ করবে আরও দ্রুত গতিতে।

কম্পিউটারের বিভিন্ন প্রজন্মকে সময়ের হিসেবে খুব চুলচেরা ভাবে ভাগ না করা গেলেও নিচের তালিকাটি থেকে বিভিন্ন প্রজন্মের 'রাজত্বকালের' মোটামুটি একটা হিসেব পাওয়া যাবে ঃ

কম্পিউটার প্রজন্ম	রাজত্বকাল
প্রথম প্রজন্ম	1940—1-52
দ্বিতীয় প্রজন্ম	1952—1964
তৃতীয় প্রজন্ম	1964—1971
চতুর্থ প্রজন্ম	1971 থেকে বর্তমান সময় পর্যন্ত

আগামী 90 এর দশকে নতুন প্রজন্মের অতি-উন্নত 'সুপার-কম্পিউটার' তৈরির জন্য বিভিন্ন কম্পিউটার প্রতিষ্ঠানগুলোর মধ্যে ইতিমধ্যেই প্রতিযোগিতা

শুরু হয়ে গেছে। 1983 সাল পর্যন্ত পৃথিবীতে মোট 60টি সুপার-কম্পিউটার ছিল। তাদের বেশিরভাগই ব্যবহার করা হয় সামরিক কাজে বা অন্যান্য কম্পিউটার তৈরি করতে। এদের মধ্যে সবচেয়ে ক্ষমতাসম্পন্ন যে যন্ত্রটি সেটি প্রতি সেকেণ্ডে 130 কোটি গণনা করতে পারে। কিন্তু এখন জাপান ও আমেরিকা যে-সব সুপার-কম্পিউটার তৈরি করতে চলেছে তারা প্রতি সেকেণ্ডে 1000 কোটিরও বেশি গণনা করতে পারবে। বিভিন্ন গবেষণায় ব্যবহারের জন্য ভারত এ-পর্যন্ত দুটি সুপার-কম্পিউটার আমদানী করার সিদ্ধান্ত নিয়েছে।

২ □ কম্পিউটার কেমন করে কাজ করে

আমাদের কাজকর্ম, চলাফেরা, কথাবার্তা সবকিছুই নিয়ন্ত্রণ করে আমাদের মস্তিষ্ক। আমাদের জ্ঞান ও অভিজ্ঞতা থেকে পাওয়া বিভিন্ন তথ্য মস্তিষ্কের স্মৃতিকোঠায় সঞ্চিত থাকে। আমরা কোনও সমস্যা চোখ বা অন্য কোনও ইন্দ্রিয় দিয়ে আঁচ করি, মস্তিষ্কে তার সঙ্কেত যায় এবং মস্তিষ্ক সেই সমস্যার সমাধান করে বিভিন্ন অঙ্গ-প্রত্যঙ্গ, মুখ, হাত-পা ইত্যাদির সাহায্যে।

আমরা যেমন বাইরের জগৎ থেকে নানান তথ্য ইন্দ্রিয়ের সাহায্যে সংগ্রহ করি, তেমনি কম্পিউটারও নানান তথ্য সংগ্রহ করে তার ইনপুট ব্যবস্থা বা ইনপুট ডিভাইস-এর মাধ্যমে। ইনপুট ব্যবস্থা কম্পিউটারের নির্দেশ বা অন্যান্য তথ্য গ্রহণ করে। এছাড়া কম্পিউটারে থাকে একটা স্মৃতিভাণ্ডার বা মেমোরি। এরই মধ্যে গণনা পদ্ধতি, সমাধান পদ্ধতি এবং অন্যান্য তথ্য সংরক্ষিত হয়।

কম্পিউটারের মস্তিষ্ক হল গণনা ও নিয়ন্ত্রণ বিভাগ, যাকে পরিভাষায় বলে প্রসেসিং অ্যাণ্ড কণ্ট্রোল ইউনিট। এই অংশ বিভিন্ন নির্দেশ বা প্রশ্নকে বিশ্লেষণ করে এবং প্রয়োজন মতো স্মৃতিতে সংরক্ষিত সমাধান ও বিভিন্ন তথ্য নিয়ে নির্দেশিত কাজ করে।

আমাদের বিভিন্ন অঙ্গ-প্রত্যঙ্গের মতো কম্পিউটারেও রয়েছে একটি আউটপুট ব্যবস্থা বা আউটপুট ডিভাইস। এর সাহায্যেই কম্পিউটার বহির্জগতের সঙ্গে যোগাযোগ রাখে।

মানুষের জটিল মস্তিষ্কের তুলনায় মানুষের সৃষ্টি কম্পিউটারের মগজ বা সি. পি. ইউ. (সেণ্ট্রাল প্রসেসিং ইউনিট) অনেক সহজ-সরল। মানুষ তার পঞ্চ ইন্দ্রিয়—চোখ, কান, নাক, জিভ ও ত্বক—দিয়ে বিভিন্ন সঙ্কেত পায় এবং পাওয়া তথ্যকে মস্তিষ্কে বিশ্লেষণ করে। আমরা ভাষা বুঝতে পারি। কোনও দৃশ্য দেখে আমাদের প্রতিক্রিয়া হয়। কিন্তু কম্পিউটার যন্ত্র—সে ওসব মোটেই বোঝে না। কম্পিউটার কাজ করে বিদ্যুৎ-প্রবাহের সাহায্যে। তার কাছে বোঝা সহজ বৈদ্যুতিক সঙ্কেত আছে কি নেই—তাই কম্পিউটারের ভাষার মূল উপাদান হল 1 অথবা 0—অর্থাৎ 'হ্যাঁ' অথবা 'না'। কম্পিউটারের ভাষা বা মেশিন ল্যাঙ্গুয়েজ তাই 1 ও 0 দিয়ে তৈরি। এখানে সব অক্ষর এবং সংখ্যা সবই 1 এবং 0 দিয়ে বোঝানো হয়। এই ভাষাকে বলে বাইনারি ভাষা।

বাইনারি সংখ্যা পদ্ধতি

আগেই বলেছি, ক্যালকুলেশন বা গণনার শুরু হয়েছিল আমাদের হাতের দশ আঙুল দিয়ে। সম্ভবত আঙুল দিয়ে সংখ্যা বোঝানোর চেষ্টা থেকেই শুরু হয়েছিল দশমিক সংখ্যার ব্যবহার। দশমিক সংখ্যার মূল বা বেস 10। উদাহরণ হিসেবে একটা সংখ্যা মনে করা যাক : 3456।

সংখ্যাটিকে একক, দশক, শতক, সহস্র এইভাবে সাজালে আমরা পাব,

সহস্র	শতক	দশক	একক
3	4	5	6

অর্থাৎ, $3456 = 3 \times 1000 + 4 \times 100 + 5 \times 10 + 6 \times 1$
$= 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 6 \times 10^0$

সুতরাং দশমিক সংখ্যায় আমরা ব্যবহার করি 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 সংখ্যাগুলো। আর এদের স্থানীয় মান অনুযায়ী দশমিক পদ্ধতির মূল 10 এর বিভিন্ন ঘাত এদের সঙ্গে গুণ করা হয়। যেমন, ওপরের উদাহরণে 4 কে গুণ করা হয়েছে 10^2 দিয়ে, অথচ 5 কে গুণ করা হয়েছে 10^1 দিয়ে। কিন্তু কম্পিউটার মাত্র দুটো সংখ্যাই বোঝে, 0 এবং 1। বাইনারি সংখ্যা পদ্ধতি এই 1 ও 0 দিয়ে গড়া। ফলে এই পদ্ধতিতে কোনও সংখ্যার মূল বা বেস 2। অর্থাৎ, এই প্রথায় দশমিক পদ্ধতির 0, 1, 2, 3, 4……ইত্যাদি দশটি সংখ্যার বদলে শুধুমাত্র দুটি সংখ্যা, 0 এবং 1, ব্যবহার করা হয়।

এখন দেখা যাক কিভাবে প্রচলিত দশমিক সংখ্যাকে বাইনারি সংখ্যায় পরিণত করা যায়। মনে করি, দশমিক সংখ্যাটি 13। এখন 13 কে ক্রমাগত 2 দিয়ে ভাগ করে যেতে হবে যতক্ষণ না ভাগফল 0 হয়।

2	13	……1 অবশেষ
2	6	……0 অবশেষ
2	3	……1 অবশেষ
2	1	……1 অবশেষ
	0	

এবার ভাগশেষগুলো এইভাবে পরপর সাজান হল : 1101

অর্থাৎ, 13 (দশমিক) = 1101 (বাইনারি)

তেমনি বাইনারি সংখ্যা থেকে দশমিক সংখ্যায় পরিবর্তনের নিয়ম হল :

$$1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 8 + 4 + 0 + 1$$
$$= 13 \text{ (দশমিক)}$$

নিচে দশমিক এবং বাইনারি সংখ্যার একটা তালিকা দেওয়া হল ঃ

দশমিক সংখ্যা	বাইনারি সংখ্যা
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

এইভাবে দশমিক পদ্ধতির যে কোনও সংখ্যাকেই 1 এবং 0 দিয়ে প্রকাশ করা যেতে পারে। একই রকম ভাবে ইংরাজি বর্ণ অথবা অন্যান্য সাঙ্কেতিক চিহ্নকেও কম্পিউটারে ব্যবহারের ক্ষেত্রে 1 ও 0 দিয়ে প্রকাশের ব্যবস্থা রয়েছে।

যেহেতু কম্পিউটার শুধু 0 এবং 1 চেনে তাই A, B, C, ইত্যাদি সমস্ত ইংরাজি অক্ষর, 0, 1, 2, 3··· ইত্যাদি সমস্ত সংখ্যা, +, −, =, ÷ ইত্যাদি বিশেষ চিহ্ন সবই 0 এবং 1 এর সমন্বয়ে তৈরি হয়ে থাকে।

প্রতিটি 0 অথবা 1 কে বাইনারি ডিজিট বা বিট বলা হয়। একসঙ্গে 4টি বা 8টি বিটকে বলে বাইট। শব্দের দৈর্ঘ্য, অর্থাৎ প্রতিটি শব্দে কতগুলি বিট আছে তাই দিয়ে নির্ধারিত হয় কম্পিউটারের কার্যক্ষমতা। 32 বিট কম্পিউটারের অর্থ ওই কম্পিউটারের শব্দ দৈর্ঘ্য 32 বিট। অর্থাৎ, একসঙ্গে 32টি বিট নিয়ে সে কাজ করতে পারে। সাধারণত 32টি বিটের মধ্যে 31টি বিট লাগে সংখ্যাকে প্রকাশ করতে, আর শেষ বিটটি কোনও সংখ্যার ধনাত্মক বা ঋণাত্মক চিহ্ন প্রকাশ করে থাকে। একে বলে পেয়ারিটি বিট।

8 বিট পদ্ধতিতে সর্বোচ্চ সংখ্যা লেখা যায় $2^8 = 256$।

8 বিট পদ্ধতিতে ইংরাজি বর্ণমালার প্রথম কয়েকটি অক্ষর হবে এইরকম ঃ

A = 10000001
B = 10000010
C = 10000011

আবার 8 বিট পদ্ধতিতে কয়েকটি দশমিক পদ্ধতির সংখ্যা হল

$$1 = 0000\ 0001$$
$$2 = 0000\ 0010$$
$$3 = 0000\ 0011$$

বাইনারি যোগ ও বিয়োগ

বাইনারি সংখ্যার সঙ্গে পরিচয়ের পর আমরা বাইনারি সংখ্যার যোগ শিখব।

বাইনারি যোগের নিয়ম হল ঃ

$$0+0=0$$
$$0+1=1$$
$$1+0=1$$
$$1+1=10$$

যোগের একটা উদাহরণ দেওয়া যাক।

দশমিক পদ্ধতি		বাইনারি পদ্ধতি
18	=	10010
+14	=	+1110
32		100000

লক্ষ্য করলে দেখা যাবে, বাইনারি যোগটির ক্ষেত্রে সব জায়গাতেই $1+1=10$ করা হয়েছে। এর মধ্যে 0 বসেছে যোগফলে এবং 1 হাতে আছে—যেটা পরের সংখ্যার সঙ্গে যোগ করা হয়েছে। এই হাতের 1 টিকে বলা হয় ক্যারি। ওপরের উদাহরণে বাইনারি যোগফল 100000 এর অর্থ দশমিক 32।

সবরকম অঙ্কের ক্ষেত্রেই কম্পিউটার অঙ্কটিকে 'যোগ'-এ রূপান্তরিত করে তবেই সমাধান করে। আর বিয়োগ করা হয় 9 এর পূরক বা নাইনস্ কমপ্লিমেণ্ট পদ্ধতির মতো। এর একটা উদাহরণ দেওয়া যাক ঃ

মূল অঙ্ক		9 এর পূরকে রূপান্তরিত অঙ্ক
8		8
−3	⇒	+6 → (9−3)
5		1 4
		+ └→1
		5

দ্বিতীয় অঙ্কে −3 এর জায়গায় +6 লেখা হল, যেহেতু 3কে 9 এর পূরকে রূপান্তরিত করা হল এইভাবেঃ (9−3)=+6

এবার দশক স্থানের 1 কে 4 এর সঙ্গে যোগ দিলেই উত্তর পাব 5।

এরকমই একটা দু-অঙ্কের বিয়োগ করে দেখা যাক।

$$\begin{array}{r} 68 \\ -42 \\ \hline 26 \end{array} \Rightarrow \begin{array}{r} 68 \\ +57 \\ \hline 125 \\ \to 1 \\ \hline 26 \end{array} \begin{array}{l} \\ \to(99-42) \\ \\ \\ \\ \end{array}$$

এখানে 42 সংখ্যাটির 9 এর পূরক 99−42=+57, এবং শতক স্থানের 1 একক স্থানের 5 এর সঙ্গে যোগ করলেই 26 সঠিক উত্তর পাওয়া যাবে।

বাইনারি পদ্ধতিতে মাত্র দুটি সংখ্যা আছে। তাই 1 এর পূরক হবে 0, এবং 0 এর পূরক হবে 1।

ওপরের উদাহরণ দুটো বাইনারির ক্ষেত্রে পরীক্ষা করে দেখা যাকঃ

দশমিক		বাইনারি		1 এর পূরকে রূপান্তরিত বাইনারি	
8	⇒	1000		1000	
−3	⇒	−0011	⇒	+1100	← পূরক বসিয়ে
5				10100	
				→ 1	যোগ
				0101	= 5 (দশমিক)

দশমিক		বাইনারি		1 এর পূরকে রূপান্তরিত বাইনারি	
68	⇒	1000100		1000100	
−42	⇒	−0101010	⇒	+1010101	← পূরক বসিয়ে
26				10011001	
				→ 1	যোগ
				0011010	= 26 (দশমিক)

আমরা অবশ্য সাধারণভাবেও বাইনারি বিয়োগ করতে পারি। বিয়োগের নিয়ম হল ঃ

$0-0=0$
$1-1=0$
$1-0=1$
$0-1=1$ (বাঁদিক থেকে 1 ধার নিতে হবে)

বিয়োগের একটা উদাহরণ নিয়ে দেখা যাক ঃ

দশমিক		বাইনারি
35	=	100011
−10	=	− 1010
25		11001 = 25 (দশমিক)

বাইনারি গুণ, ভাগ ও ভগ্নাংশ

এবারে বাইনারি গ‌ুণ ও ভাগ কিভাবে করে সেটা দেখা যাক।

গ‌ুণের নিয়ম হল ঃ

$0\times0=0$
$1\times0=0$
$1\times1=1$
$0\times1=0$

একটা উদাহরণ দেওয়া যাক ঃ

দশমিক		বাইনারি
35	=	100011
×10	=	×001010
350		000000
		100011×
		000000×
		100011×
		101011110 = 350 (দশমিক)

বাইনারি ভাগের নিয়ম হল ঃ

$$0 \div 1 = 0$$
$$1 \div 1 = 1$$
$$0 \div 0 = \text{অর্থহীন}$$
$$1 \div 0 = \text{অর্থহীন}$$

বাইনারি ভাগও সাধারণ ভাগের মতো। এর কোনও উদাহরণ দেওয়া হল না। পাঠকের ওপরে ছেড়ে দেওয়া হল।

এবারে বাইনারি ভগ্নাংশের একটা উদাহরণ দেওয়া যাক। বাইনারি 0·011 এর অর্থ হল ঃ

$$\begin{aligned} 0{\cdot}011 &= 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} \\ &= 0 \times \tfrac{1}{2} + 1 \times \tfrac{1}{4} + 1 \times \tfrac{1}{8} \\ &= 0 + 1 \times {\cdot}25 + 1 \times {\cdot}125 \\ &= 0{\cdot}375 \ (\text{দশমিক}) \end{aligned}$$

আগেই বলেছি, কম্পিউটার গুণ করে পরপর যোগ করে। অর্থাৎ 5 কে 8 দিয়ে গুণ করতে হলে ধাপগুলো হবে এই রকম ঃ

$$8+8=16, \quad 16+8=24, \quad 24+8=32, \quad 32+8=40$$

তেমনি ভাগ করতে হলে পরপর বিয়োগ করে যেতে হবে। যেমন, 40 কে 8 দিয়ে ভাগ করার পদ্ধতি হল ঃ

$$40-8=32, \; 32-8=24, \; 24-8=16, \; 16-8=8, \; 8-8=0$$

অর্থাৎ, মোট 5টি ধাপ লাগবে।

গুণ ও ভাগের পদ্ধতির ধাপের সংখ্যা দেখে ভয় পাওয়া স্বাভাবিক—বিশেষত যেখানে অঙ্ক খুব জটিল ও দীর্ঘ। কিন্তু কম্পিউটারের অত্যন্ত দ্রুত গতির কাছে গুণ-ভাগের অসংখ্য ধাপও কিছুই নয়।

কম্পিউটারের স্মৃতি ও গঠন

কম্পিউটারকে কিভাবে কাজ করতে হবে, তার নির্দেশ এবং অন্যান্য তথ্য বাইনারি সংখ্যায় রূপান্তরিত করে কম্পিউটারের স্মৃতিতে রাখা হয়। দরকার মতো কম্পিউটার ওই সব তথ্য এবং নির্দেশ পাঠ করে কাজ করে। সুতরাং কম্পিউটারের স্মৃতিতে বাইনারি সংখ্যা 'লেখা' এবং 'পড়ার' ব্যবস্থা থাকা প্রয়োজন। কম্পিউটারের স্মৃতি অসংখ্য কোষ দ্বারা গঠিত। এরকম এক একটি

কোষে এক একটি বিট সংরক্ষিত হতে পারে। কোনও কম্পিউটারের ক্ষমতা তার স্মৃতিকোষের সংখ্যার ওপর নির্ভর করে। 64K কম্পিউটারের অর্থ তার স্মৃতিতে $64 \times 1024 = 65536$ সংখ্যক স্মৃতিকোষ আছে। কারণ,

$$1K = 2^{10} = 1024$$

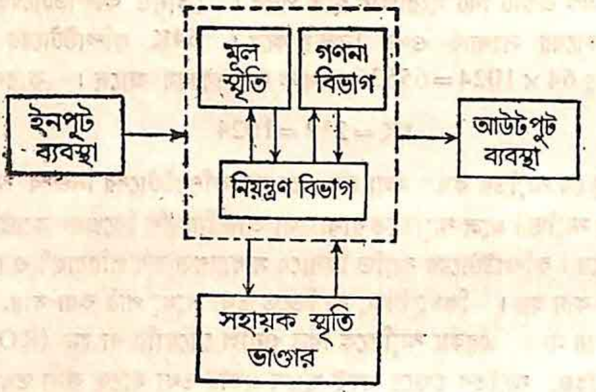
এই যে স্মৃতির কথা বলা হল, তা হল কম্পিউটারের নিজস্ব স্মৃতিভাণ্ডার বা মূল স্মৃতি। মূল স্মৃতিতে রাখা তথ্য আর নির্দেশ বিশ্লেষণ করেই কম্পিউটার কাজ করে। কম্পিউটারের স্মৃতি হিসাবে সাধারণত অর্ধপরিবাহী ও চৌম্বক বস্তু ব্যবহার করা হয়। কিছু কিছু স্মৃতিতে তথ্য শুধু পাঠ করা যায়, নতুন তথ্য লেখা যায় না। এরকম স্মৃতিকে রিড ওনলি মেমোরি বা রম (ROM) বলে। কিছু কিছু স্মৃতির ক্ষেত্রে একই স্থানে একটি তথ্য মুছে অন্য তথ্য লেখা যায় এবং নতুন লেখা তথ্য সংরক্ষণ করা যায়। এরকম স্মৃতিকে র‍্যাণ্ডম অ্যাকসেস মেমোরি বা র‍্যাম (RAM) বলে।

মানুষ নিত্য যে কাজকর্ম করে তার সব খুঁটিনাটি সবসময় সে মনে রাখতে পারে না। অর্থাৎ তার মস্তিষ্কের সীমিত স্মৃতির ক্ষমতার ওপরে সবসময় সে ভরসা করতে পারে না। তাই সে তার নিজের অভিজ্ঞতা, পূর্বসূরির অভিজ্ঞতা ইত্যাদি লিখে রাখে কাগজপত্রে বা বইখাতায়। প্রয়োজন মতো মানুষ সে-সব তথ্য-ভাণ্ডারের সাহায্য নেয়। মানুষের মতো কম্পিউটারের মূল স্মৃতি বা মেইন মেমোরিও সীমিত। তাকে সাহায্য করার জন্য থাকে সহায়ক স্মৃতি-ভাণ্ডার বা ব্যাক-আপ স্টোরেজ। এর কাজ হল যাবতীয় তথ্য স্থায়ীভাবে সংরক্ষিত করা, যাতে প্রয়োজন হলেই সেই তথ্য কম্পিউটারের মূল স্মৃতিতে নিয়ে আসা যায় এবং প্রয়োজনীয় গণনার কাজ করা যায়।

এবারে আমরা কম্পিউটারের গঠন নিয়ে দু-চার কথা বলব। এ থেকেই বোঝা যাবে চার্লস ব্যাবেজ পরিকল্পিত গঠনের সঙ্গে আধুনিক কম্পিউটারের গঠনের কতটা মিল।

আমরা দেখেছি যে কম্পিউটারে নিম্নলিখিত মূল অংশগুলি থাকে :

১। ইনপুট ব্যবস্থা
২। কেন্দ্রীয় ক্রিয়া বিভাগ বা সেণ্ট্রাল প্রসেসিং ইউনিট (সি. পি. ইউ.) ঃ
এটি আবার তিনটি অংশে বিভক্ত ঃ
(ক) নিয়ন্ত্রণ বিভাগ বা কন্ট্রোল ইউনিট
(খ) গণনা বিভাগ বা এরিথমেটিক অ্যাণ্ড লজিক ইউনিট
(গ) মূল স্মৃতি বা মেইন মেমোরি
৩। আউটপুট ব্যবস্থা
৪। সহায়ক স্মৃতিভাণ্ডার



আধুনিক কম্পিউটারের গঠনের ছবি

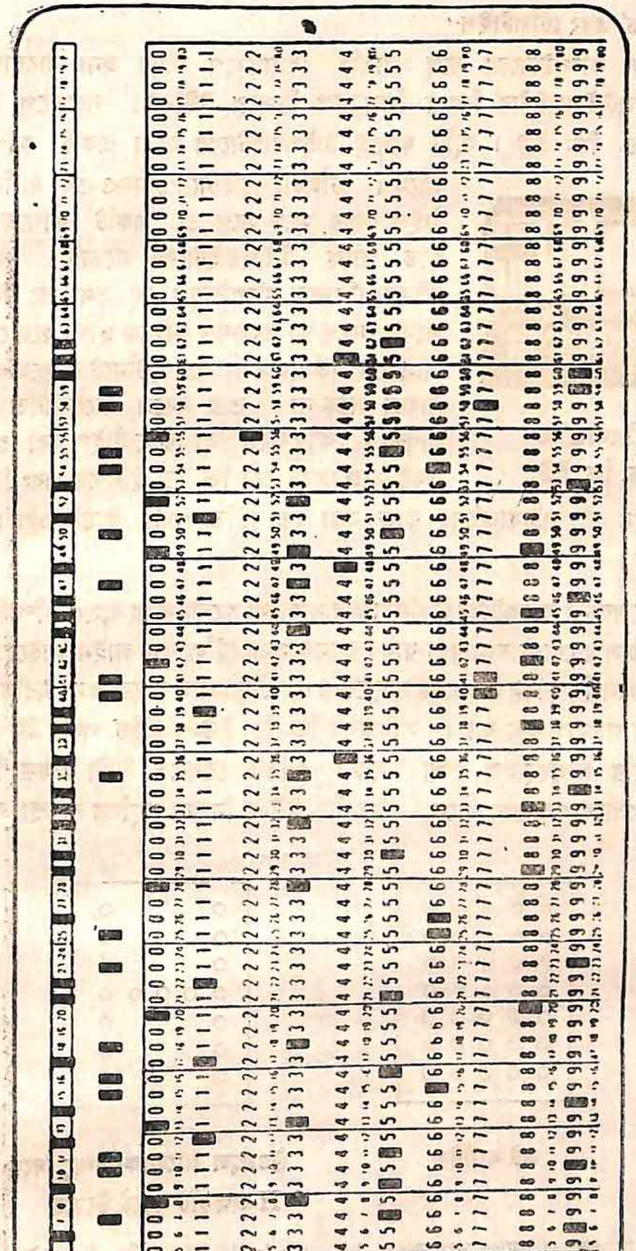
বহির্জগতের সঙ্গে কম্পিউটারের যোগাযোগের জন্য, অর্থাৎ তথ্য বা ডেটা আদান-প্রদানের জন্য, বিভিন্ন ব্যবস্থা আছে। ইনপুট ব্যবস্থার মাধ্যমে আমরা তথ্য এবং নির্দেশ কম্পিউটারের ভেতরে পাঠিয়ে দিই এবং গণনা বা বিশ্লেষণের ফলাফল আউটপুট ব্যবস্থার মাধ্যমে জানতে পারি।

এবারে কম্পিউটারের মূল অংশগুলি নিয়ে সংক্ষেপে আলোচনা করা যাক।

ইনপুট ব্যবস্থা

কার্ড রীডার

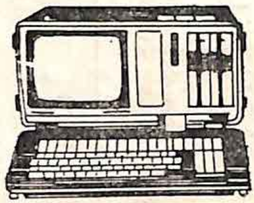
কয়েক বছর আগেও পাঞ্চ কার্ড ব্যবস্থা ব্যাপকভাবে ব্যবহার করা হত। একটা কার্ডের বিভিন্ন জায়গায় ছিদ্র করে বিভিন্ন বর্ণ এবং সংখ্যাকে চিহ্নিত করা হয়। তারপর সেই ছিদ্রযুক্ত কার্ডটিকে কম্পিউটারের কার্ড রীডারে দেওয়া হয়। কার্ড রীডারের কাজ তার নামেই স্পষ্ট। সে কার্ড 'রীড' করে বা পাঠ করে। অর্থাৎ, সে কার্ডে ছিদ্রগুলির উপস্থিতি এবং অবস্থান অনুভব করে বর্ণ বা সংখ্যাটিকে বুঝতে পারে এবং 1 এবং 0 দিয়ে তৈরি যান্ত্রিক ভাষায় কম্পিউটারকে সম্পূর্ণ নির্দেশটি জানিয়ে দেয়। কার্ড পাঞ্চিং যন্ত্রে অপারেটর টাইপ মেশিনের মতো বোতাম টেপে, আর তাতেই চিহ্ন বা সংকেত অনুযায়ী ছিদ্র হয়ে যায় কার্ডে। পরে সেই কার্ড কার্ড রীডারের মাধ্যমে সঙ্কেত পাঠায় কম্পিউটারে। কার্ড একবার পাঞ্চ করে ব্যবহার করা হয়ে গেলে তাতে নতুন নির্দেশ আর পাঞ্চ করা সম্ভব হয় না। সেই জন্যই বাতিল কার্ড প্রাইভেট বাসের টিকিট ছাপার কাজে ব্যবহার করা হয়।



পাঞ্চড্ কার্ড। কালো ছোট ছোট আয়তাকার অংশগুলো কার্ডের ছিদ্র বোঝাচ্ছে।

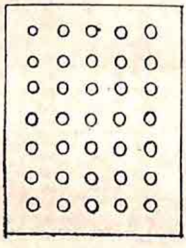
কী-বোর্ড এবং টেলিটাইপ

আজকাল কম্পিউটারের সঙ্গে সরাসরি যোগাযোগ করার জন্য ব্যবহার করা হয় টেলিটাইপরাইটার কিংবা ভিস্যুয়াল ডিসপ্লে ইউনিট। সংক্ষেপে একেই বলে ভি. ডি. ইউ.। দুটি যন্ত্রেই টাইপরাইটারের মতো একটি কী-বোর্ড আছে। টেলিটাইপরাইটারে কোনও তথ্য বা নির্দেশ টাইপ করার সঙ্গে সঙ্গে তা একটি কাগজে ছাপা হতে থাকে টাইপরাইটারের মতোই। অর্থাৎ, কী-বোর্ড থেকে কম্পিউটারে যে তথ্য বা নির্দেশ প্রবেশ করছে তা তৎক্ষণাৎ কাগজে ছাপা হয়ে বেরিয়ে আসছে। ভিস্যুয়াল ডিসপ্লে ইউনিটে কাগজে ছাপার ব্যবস্থা থাকে না। তার বদলে থাকে টিভি পর্দার ব্যবস্থা। কী-বোর্ডে যা কিছু টাইপ করা হয় তা একই সঙ্গে ফুটে ওঠে ভি. ডি. ইউ.-তে এবং নির্দেশ সংশোধন বা পরিমার্জনের কাজ করা হয় টিভি পর্দায় ফুটে ওঠা লেখার ওপরেই।

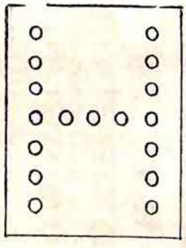


কী-বোর্ড ও
ভি. ডি. ইউ.

এই ধরনের কম্পিউটার টারমিনালের সাহায্যে অনেক দূরে বসেও কম্পিউটারের সঙ্গে সরাসরি যোগাযোগ করা যায়। অনেক সময় টেলিফোন লাইন, মাইক্রোওয়েভ লিঙ্ক, এমন কি কৃত্রিম উপগ্রহের সাহায্যেও এক টারমিনাল থেকে অন্য টারমিনালের সঙ্গে যোগাযোগ করা যায়। সাধারণত ভি. ডি. ইউ.-র টিভি পর্দায় 20 থেকে 40 সারি অনুভূমিক রেখা থাকে। প্রতিটি রেখাতে 80টা অক্ষর, চিহ্ন বা সংখ্যা বসানোর জায়গা থাকে। ভি. ডি. ইউ.-র নিজস্ব স্মৃতির ব্যবস্থা আছে।



ডট ম্যাট্রিক্স



উজ্জ্বল আলোক-বিন্দু দিয়ে
H-অক্ষরটি ফুটে উঠেছে

ওই স্মৃতির পরিমাপ সাধারণত 1K, অর্থাৎ 1024। ভি. ডি. ইউ.-র পর্দায় সাধারণত ডট্ ম্যাট্রিক্স পদ্ধতিতে অক্ষর বা সংখ্যা লেখা হয়ে থাকে। এতে 5×7, অর্থাৎ 35টি আলোকিত বিন্দু দিয়ে বর্ণ বা সংখ্যা গঠন করা হয়।

ফলে কোনও বিশেষ অক্ষর লিখতে চাইলে সেই অক্ষরের জন্য প্রয়োজনীয় বিন্দুগুলো ভাস্বর হয়ে উঠে অক্ষরটি গঠন করে। এই ভি. ডি. ইউ.-কে ইনপুটের মতো আউটপুট হিসেবেও ব্যবহার করা যায়।

এছাড়া কিছু বিশেষ ধরনের ভি. ডি. ইউ. টারমিনাল আছে যাদের সাহায্যে বিভিন্ন জ্যামিতিক নক্সা বা গ্র্যাফিক্‌স আঁকা যায় এবং তা কম্পিউটারকে বোঝানো যায়। কম্পিউটার ওই নক্সার বিভিন্ন অংশের অবস্থান তার স্মৃতিতে সংরক্ষিত করে রাখে এবং প্রয়োজনীয় বিশ্লেষণের কাজ করে।

কেন্দ্রীয় ক্রিয়া বিভাগ বা সি. পি. ইউ.

আমরা আগেই দেখেছি, কেন্দ্রীয় ক্রিয়া বিভাগ তিনটে বিভাগের সমন্বয়ে গঠিত। তারা হল গণনা বিভাগ, নিয়ন্ত্রণ বিভাগ এবং মূল স্মৃতি। গণনা বিভাগের মূল কাজ হল যোগ, বিয়োগ, গুণ, ভাগ—এই সব অঙ্ক করা, কিছু যুক্তিগ্রাহ্য কার্যকলাপ সম্পন্ন করা, তথ্য বিশ্লেষণ করা এবং সিদ্ধান্ত নেওয়া।

উদাহরণ হিসেবে ধরা যাক, যে কোনও ছাত্রের শতকরা মোট নম্বর 60% এর বেশি হলে তাকে প্রথম বিভাগে উত্তীর্ণ ঘোষণা করা হবে, 50% এর বেশি হলে দ্বিতীয় বিভাগে এবং 40% এর বেশি হলে তৃতীয় বিভাগে সে উত্তীর্ণ হবে। আর নম্বর শতকরা 40 এর কম হলে ছাত্রটিকে অকৃতকার্য ঘোষণা করা হবে। এই সমস্যার সমাধান করতে হলে গণনা বিভাগ প্রথমে ছাত্রের মোট নম্বর যোগ করবে। শতকরা হার নির্ণয় করবে। তারপর 60%, 50%, 40% এর সঙ্গে সেই শতকরা হারের তুলনা করবে। আর সবশেষে যুক্তিগ্রাহ্য সিদ্ধান্ত গ্রহণ করবে। সাধারণত বিভিন্ন ইলেকট্রনিক লজিক বর্তনীর সাহায্যে গণনা বিভাগ এই যুক্তিনিষ্ঠ বিশ্লেষণ করে থাকে।

কম্পিউটারের যাবতীয় নির্দেশ এবং তথ্য রাখা থাকে তার মূল স্মৃতিতে। পর্যায়ক্রমে কম্পিউটারকে কী করতে হবে সেখানে তার নির্দেশ থাকে এবং যে-সব তথ্য নিয়ে কম্পিউটারকে কাজ করতে হবে তার ঠিকানা সংরক্ষিত থাকে। কম্পিউটারের নিয়ন্ত্রণ বিভাগ মূল স্মৃতি থেকে একের পর এক নির্দেশ পাঠ করে এবং সেই নির্দেশ অনুযায়ী গণনা বিভাগকে অথবা ইনপুট-আউটপুট ব্যবস্থাকে কাজের নির্দেশ দেয়। কোনও তথ্যের ঠিকানা জেনে সেই তথ্যকে মূল স্মৃতি থেকে উদ্ধার করে পাঠ করে নিয়ন্ত্রণ বিভাগ। তারপর সে গণনা বিভাগ অথবা আউটপুট বিভাগকে সেই তথ্য সরবরাহ করে। আবার গণনা বিভাগ থেকে পাওয়া গণনার ফলাফল, অথবা, ইনপুট বিভাগ থেকে পাওয়া তথ্য নির্দেশ অনুযায়ী মূল স্মৃতির বিশেষ জায়গায় সংরক্ষণ করা, কিংবা তা আউটপুট বিভাগের মাধ্যমে প্রকাশ করা—এর দায়িত্বও নিয়ন্ত্রণ বিভাগের।

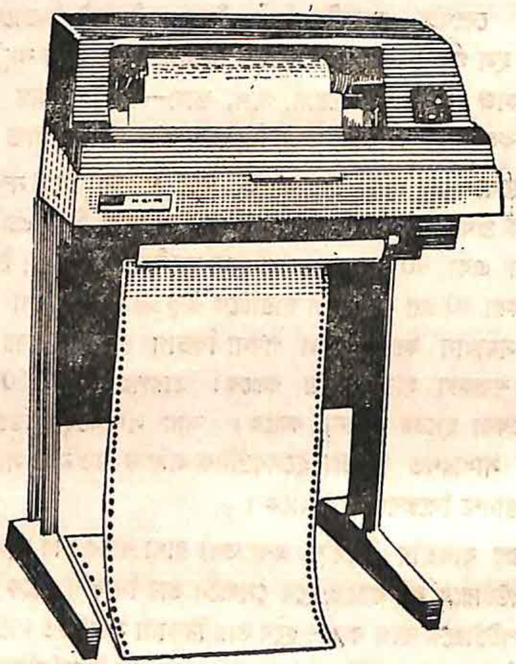
এক কথায় কম্পিউটারের মধ্যে তথ্য আদান-প্রদান এবং তথ্যের প্রবাহের গতিপথের সামগ্রিক পরিচালনার দায়িত্ব নিয়ন্ত্রণ বিভাগের।

আউটপুট ব্যবস্থা

সবচেয়ে প্রচলিত আউটপুট ব্যবস্থার নাম প্রিন্টার বা মুদ্রণ-যন্ত্র। প্রিন্টার সাধারণত দুরকমের হয়। লাইন প্রিন্টার এবং ডট ম্যাট্রিক্স প্রিন্টার।

লাইন প্রিন্টার

লাইন প্রিন্টারে বিভিন্ন ধরনের বর্ণ ও চিহ্ন একটা ঘূর্ণায়মান চোঙের গায়ে অথবা একটা বৃত্তাকার ফিতের মধ্যে সারিবদ্ধভাবে সাজানো থাকে।

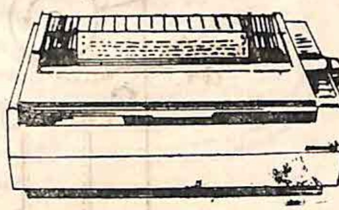


লাইন প্রিন্টার

কম্পিউটারের বিশ্লেষণ অথবা কোনও সমস্যার সমাধান প্রিন্টারের সাহায্যে ছাপতে চাইলে কম্পিউটার তার প্রত্যেকটি বর্ণ বা চিহ্নের বাইনারি সঙ্কেত প্রিন্টারে পাঠায়। তখন প্রিন্টারে অবস্থিত বর্ণমালা উপযুক্ত বর্ণ বা চিহ্নকে কালির রিবনের সাহায্যে ছেপে দেয় কাগজে। এই কাজটি হয় খুব দ্রুত। ফলে এতে একেবারে একটা গোটা লাইন ছাপা হয়ে যায় কাগজে। সাধারণত লাইন প্রিন্টার মিনিটে 200 থেকে 2000 লাইন পর্যন্ত ছাপতে পারে।

ডট ম্যাট্রিক্স প্রিণ্টার

এই ধরনের প্রিন্টারের ছাপার গতি লাইন প্রিণ্টারের মতো অত দ্রুত নয়। কিন্তু এর সুবিধেজনক আকার এবং কম দামের জন্য এই প্রিণ্টার বহুল প্রচলিত। এতে এক-একবারে একটা করে বর্ণ ছাপা হয়—অনেকটা সাধারণ টাইপরাইটারের মতো।


ডট ম্যাট্রিক্স প্রিণ্টার

ডট ম্যাট্রিক্স প্রিণ্টারে প্রতিটি বর্ণ বা সংখ্যা কতকগুলো পিনের সাহায্যে বিন্দু চিহ্ন বা ফুটকি দিয়ে আঁকা হয়ে যায় কাগজের ওপরে—ঠিক ভি. ডি. ইউ.-র টিভি পর্দার ডট ম্যাট্রিক্সের মতো। সাধারণত 35টি (5 × 7) ডট ম্যাট্রিক্স পিন ব্যবহার করা হয় এই প্রিণ্টারে। যে বর্ণ বা সংখ্যা লিখতে হবে পিনগুলো সেই অনুযায়ী নিজেদের সাজিয়ে নেয়। তারপর বিশেষ ধরনের কালির সাহায্যে কাগজে চিহ্ন পড়ে এবং সেই বর্ণ বা সংখ্যাটি কাগজে ফুটে ওঠে।

প্রিণ্টার ছাড়া আউটপুট ব্যবস্থায় ব্যবহার করা যায় কার্ড পাঞ্চ যন্ত্র বা ভি. ডি. ইউ.। এ ছাড়া কম্পিউটারের সাহায্যে জ্যামিতিক নক্সা আঁকার জন্য প্লটারও ব্যবহার করা হয়। এতে কাগজের ওপরে পেনের চলাচল নিয়ন্ত্রিত হয় কম্পিউটারের সাহায্যে এবং প্রয়োজনীয় বিভিন্ন নক্সা অতি সহজে আঁকা হয়ে যায়।

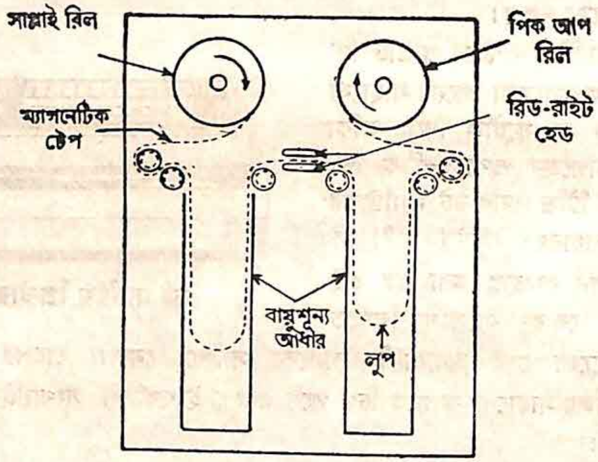
সহায়ক স্মৃতিভাণ্ডার

কম্পিউটারের নিজস্ব স্মৃতিভাণ্ডারের সীমিত ক্ষমতার জন্য সহায়ক স্মৃতি-ভাণ্ডারের ব্যবস্থা থাকে। এর কাজ হল যাবতীয় তথ্য স্থায়ীভাবে সংরক্ষণ করা, যাতে প্রয়োজনে সেই তথ্য সি. পি. ইউ.-র মাধ্যমে মূল স্মৃতিতে আনা যায় এবং প্রয়োজনীয় বিশ্লেষণ করা যায়। এরকম সহায়ক স্মৃতিভাণ্ডার হিসেবে ম্যাগনেটিক টেপ, ম্যাগনেটিক ডিস্ক, ফ্লপি ডিস্ক ইত্যাদি উল্লেখযোগ্য।

ম্যাগনেটিক টেপ

চৌম্বক ফিতে বা ম্যাগনেটিক টেপ সাধারণ টেপ রেকর্ডারে ব্যবহৃত টেপের মতোই। অর্থাৎ, সাধারণ টেপ-এ যেমন গান বা কণ্ঠস্বর রেকর্ড করা হয় এবং দরকার মতো তা পুনরুদ্ধার করা যায়, বা বাজানো যায়, তেমনি কম্পিউটারের ম্যাগনেটিক টেপেও বাইনারি ভাষায় তথ্য সংরক্ষণ করা যায় এবং প্রয়োজনে পুনরুদ্ধার করা যায়। সাধারণত এই ধরনের টেপ বা ফিতে আধ ইঞ্চি চওড়া

এবং 1200 ফুট থেকে 2400 ফুটের মতো লম্বা হয়। এই টেপ চালনা করে যে যন্ত্র তাকে বলা হয় ম্যাগনেটিক টেপ ড্রাইভ। ম্যাগনেটিক টেপের একটা সাপ্লাই রিল এবং একটা পিক্ আপ রিল থাকে। আর টেপের চলাচল বায়ুশূন্য আধারের মধ্যে করা হয়ে থাকে।



ম্যাগনেটিক টেপ ড্রাইভ

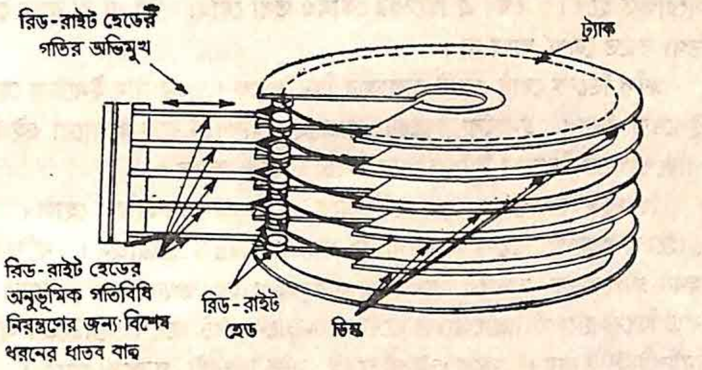
আধুনিক টেপে এক ইঞ্চিতে প্রায় 1600 অক্ষর বা চিহ্ন ধরে রাখা যায়। ম্যাগনেটিক টেপে তথ্য সংরক্ষণ করা বা সংরক্ষিত তথ্য উদ্ধার করার কাজ করে রিড-রাইট হেড। অর্থাৎ রিড-রাইট হেড টেপের ওপর কোনও তথ্য 'লিখতে' পারে বা লেখা কোনও তথ্য 'পড়ে ফেলতে' পারে। তবে ম্যাগনেটিক টেপের একটা অসুবিধে আছে। এ থেকে চটপট তথ্য পাওয়ার উপায় নেই—কারণ টেপ চালু করে ঐ তথ্যের নির্দিষ্ট জায়গাটি রিড রাইট হেডের কাছে এলে তবেই তথ্যটি পড়া যাবে। যদি কোনও তথ্য টেপের মাঝামাঝি কোনও জায়গায় থাকে তবে তা আহরণের সময় লাগবে অনেক বেশি।

ম্যাগনেটিক ডিস্ক

চৌম্বক চাকতি বা ম্যাগনেটিক ডিস্ক বিভিন্ন আকারের হয়। এই ডিস্কের ওপর গ্রামোফোন রেকর্ডের মতো বিভিন্ন ট্র্যাক থাকে। তবে এই ট্র্যাক চুম্বকীয়। একটি ডিস্কে প্রায় 200 টি ট্র্যাক থাকে। আর ডিস্কের ওপরে ও নিচে দু' পিঠেই তথ্য থাকে।

ছবির ম্যাগনেটিক ডিস্কে ওপর নিচ মিলিয়ে মোট বারোটি তল আছে। তার মধ্যে দশটি তলে তথ্যকে ধরে রাখা যায়। এই ডিস্ক থেকেও রিড-রাইট হেড-

এর মাধ্যমে তথ্য আদান-প্রদান করা হয়। তবে দশটি তল থেকে দ্রুত তথ্য পড়া বা লেখার জন্য এখানে দশটি রিড-রাইট হেড ব্যবহার করা হয়েছে।

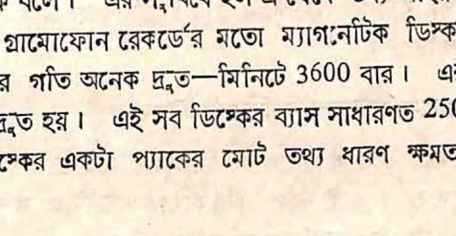


ম্যাগনেটিক ডিস্ক

এই ধরনের ডিস্ককে হার্ড ডিস্ক বলে। এর সুবিধে হল এ থেকে তথ্য আহরণ করতে সময় লাগে খুব কম। গ্রামোফোন রেকর্ডের মতো ম্যাগনেটিক ডিস্ক-গুলোও ঘোরে। তবে এদের গতি অনেক দ্রুত—মিনিটে 3600 বার। এর ফলে তথ্য আহরণের কাজটিও দ্রুত হয়। এই সব ডিস্কের ব্যাস সাধারণত 250 মিলিমিটার হয়। 12টি ডিস্কের একটা প্যাকের মোট তথ্য ধারণ ক্ষমতা 3000 মেগাবাইট।

ফ্লপি ডিস্ক

বর্তমানে সবচেয়ে জনপ্রিয় এবং বহুল ব্যবহৃত তথ্য সংরক্ষণ ব্যবস্থা হল ফ্লপি ডিস্ক বা ডিস্কেট। এগুলো নমনীয় প্লাস্টিকের পাত দিয়ে তৈরি। সাধারণত এদের ব্যাস 130 মিলিমিটার (সোয়া পাঁচ ইঞ্চি) বা 200 মিলিমিটার (আট ইঞ্চি) হয়। প্রতিটি ফ্লপি ডিস্ক একটা খামে ভরা থাকে। এই খামে ক্যাপসুলের চেহারার একটা গর্ত কাটা থাকে। এই গর্তের মাধ্যমেই ডিস্কের ট্র্যাক রিড-রাইড হেডের সংস্পর্শে আসে।



ফ্লপি ডিস্ক

ফ্লপি ডিস্কের এক ধারে একটা ঘাট বা নচ্ কাটা থাকে। এই নচ্

খোলা থাকলে ডিস্কে তথ্য লেখা যায় এবং মোছা যায়। কিন্তু ঐ নচের জায়গায় আঠা দিয়ে এক টুকরো টেপ লাগিয়ে দিলে ডিস্কের তথ্য পুরোপুরি সংরক্ষিত হবে। তখন ঐ ডিস্কের কোনও তথ্য মোছা যাবে না বা নতুন কোনও তথ্য তাতে লেখা যাবে না।

ফ্লপি ডিস্কে ছোট্ট একটি বৃত্তাকার ছিদ্র থাকে। এর নাম ইনডেক্স হোল বা ইনডেক্স ছিদ্র। ইনডেক্স ছিদ্রের কাজ হল কম্পিউটারকে জানানো ওই ছিদ্রের পরিপ্রেক্ষিতে ডিস্কের বিভিন্ন ট্র্যাক কতটা দূরত্বে আছে।

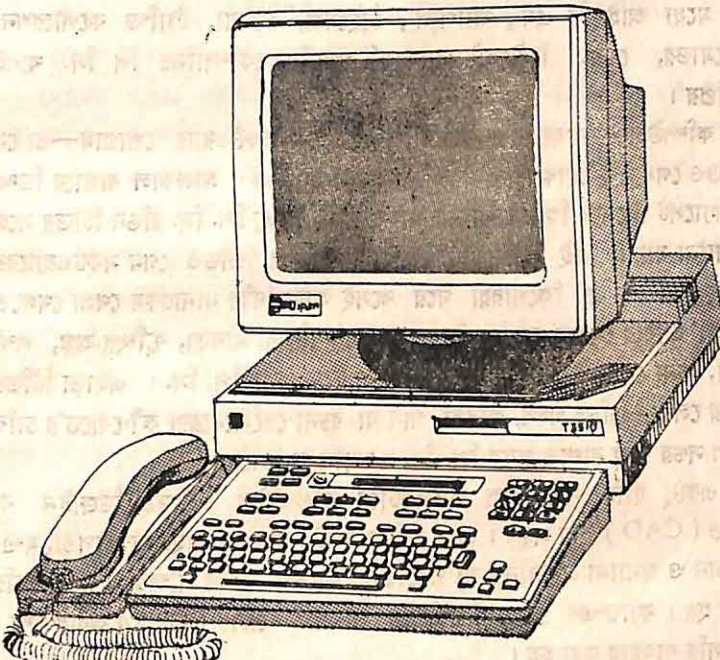
ডিস্কের কেন্দ্রে যে বড় ছিদ্রটি থাকে তার নাম সেন্টারিং হোল। ডিস্ক ড্রাইভের অক্ষদণ্ড ডিস্কে প্রবেশ করার জন্য এই ছিদ্রটি প্রয়োজন। কম্পিউটারে যখন ফ্লপি ডিস্ক ঢুকিয়ে দিয়ে কাজ শুরু করা হয়, তখন ডিস্ক ড্রাইভের অক্ষ-দণ্ড ডিস্কের সেণ্টারিং হোলের মধ্যে ঠিকভাবে এঁটে বসে। ড্রাইভের আবর্তন মিনিটে 300 বার। ফলে ওই গতিতেই ফ্লপি ডিস্কটা ঘুরতে থাকে। একটা 200 মিলিমিটার ফ্লপি ডিস্কের দুটি তলের ওপরে প্রায় 2 মেগাবাইট তথ্য ধরে রাখা যায়। ডিস্কে তথ্য রেকর্ড করা এবং পুনরুদ্ধার করা হয় চুম্বকীয় পদ্ধতিতে।

পার্সোনাল কম্পিউটার

আমরা আগেই দেখেছি সিলিকন মাইক্রোচিপস্ কম্পিউটার-শিল্পে এনেছে যুগান্তর। এরই ফলে তৈরি হয়েছে পার্সোনাল কম্পিউটার বা হোম কম্পিউটার। পি. সি. বা পার্সোনাল কম্পিউটার উদ্ভাবনের ফলে এবং ঘরে ঘরে এর ব্যাপক ব্যবহারের জন্য সাধারণ মানুষ কম্পিউটারের খুব কাছে আসতে পেরেছে। একটা ছোট পি. সি. বা হোম কম্পিউটার শিশু বা কিশোরদের কাছে তো বটেই, এমন কি পরিবারের সকলের কাছেই আকর্ষণীয়। এটা দিয়ে স্কুল-কলেজের সবরকম হোমওয়ার্ক চটপট করে ফেলা যায়। শিল্পপতিদের কাছে পি. সি. ব্যবসা-বাণিজ্যের ভবিষ্যৎ পরিকল্পনার কাজে লাগে। এছাড়া ব্যক্তিগত হিসেব রাখার কাজেও পি. সি.-র তুলনা নেই।

একটা পি. সি. বা হোম কম্পিউটারে বড় কম্পিউটারের মতো সি. পি. ইউ., ইনপুট-আউটপুট ব্যবস্থা, স্মৃতি বা মেমোরি সব কিছুই আছে—তবে তার সবই আয়তনে ছোট। যখন কম্পিউটারের কার্যকর স্মৃতি 8 কিলোবাইট থেকে 16, 32 বা 64 কিলোবাইট হল, তখন কম্পিউটার পেল অতিরিক্ত গতি এবং কার্যক্ষমতা। সাধারণ পার্সোনাল কম্পিউটারে 64K কার্যকর স্মৃতি থাকে। আর এই কম্পিউটার যে-দ্রুততায় কোনও কাজ করতে পারে তা আমাদের নিত্য প্রয়োজনীয় গণনা বা কাজের পক্ষে যথেষ্ট। তাই দামে সস্তা পার্সোনাল কম্পিউটার আজ খুবই জনপ্রিয়। কোনও কোনও পি. সি.-তে এল. সি. ডি.

(লিকুইড ক্রিস্টাল ডিসপ্লে) পর্দা থাকে। আবার একটু বড় ধরনের পি. সি -তে ভি. ডি. ইউ. থাকে। আর ইনপুট ব্যবস্থা হিসেবে সকলেরই থাকে কী-বোর্ড।



পার্সোনাল কম্পিউটার

আমেরিকার টাইমেক্স কোম্পানির তৈরি 'সিনক্লেয়ার ওয়ান থাউজ্যান্ড' একটা ছোট পি. সি.-র উদাহরণ। এর সঙ্গে আমাদের বাড়ির সাদা-কালো বা রঙিন টেলিভিশন সেট আউটপুট ব্যবস্থা হিসেবে যোগ করে নেওয়া যায়। সঙ্গে সাহায্যকারী স্মৃতি হিসেবে লাগানো যায় নিত্য ব্যবহৃত পোর্টেবল ক্যাসেট টেপ রেকর্ডার। সিনক্লেয়ার ওয়ান থাউজ্যান্ড-এর আয়তন দৈর্ঘ্যে আট ইঞ্চি, প্রস্থে ছয় ইঞ্চি এবং উচ্চতায় দু-ইঞ্চির বেশি নয়। এই আয়তনই হল আসল সি. পি. ইউ, মূল স্মৃতি এবং ইনপুট কী-বোর্ডের-সমষ্টি। এর সাহায্যে বাড়িতে ব্যবহৃত টিভির পর্দায় শিশুদের প্রিয় ভিডিও গেমস্-এর যাবতীয় প্রোগ্রাম, স্কুলের হোমওয়ার্ক, এমন কী বাণিজ্যিক এবং অর্থনৈতিক গণনার কাজও করা যায়। এরকমই কয়েকটা উদাহরণ দেওয়া যাক। কম্পিউটার কোনও ব্যক্তির বন্ধু-বান্ধব, আত্মীয়-স্বজনের নাম, জন্ম-তারিখ, বিবাহ-বার্ষিকীর তারিখ ইত্যাদি

দিব্যি মনে রাখতে পারে। এছাড়া কারও আয়-ব্যয়ের হিসাব, এল. আই. সি. পলিসির প্রিমিয়াম, ব্যাঙ্কের অ্যাকাউণ্টের খবর, বিভিন্ন সভা-সমিতির চাঁদা— এরকম আরও কত কি মনে রাখতে পারে পি. সি.। তাই অল্প দামের পি. সি. এত জনপ্রিয়। শুধু আমেরিকাতেই একশোর ওপর পি. সি. নির্মাতা রয়েছে। এর মধ্যে আই. বি. এম., অ্যাপল্, টাইমেক্স, ওমেগা, ট্যাণ্ডি কর্পোরেশন, কোমোডর, ডেক, হিউলেট প্যাকার্ড প্রভৃতি কোম্পানির পি. সি. খুবই জনপ্রিয়।

কম্পিউটারে ব্যবহারের উপযোগী বিভিন্ন সফটওয়্যার প্রোগ্রাম—তা সে ভিডিও গেম্স্ই হোক আর দরকারী হিসেবই হোক—আজকাল বাজারে ডিস্ক বা ক্যাসেট হিসেবে কিনতে পাওয়া যায়। কিছু কিছু পি. সি. রঙিন টিভির সঙ্গে লাগানো যায়। এই রঙিন টিভির পর্দায় কালার ভিডিও গেম সফটওয়্যারের সাহায্যে শিশু ও কিশোররা ঘরে বসেই আকর্ষণীয় নানারকম খেলা খেলতে পারে। আবার খেলার মাধ্যমে শিশুদের বর্ণ পরিচয়, নামতা, বুদ্ধির অঙ্ক, শব্দ গঠন, রচনা লেখা ইত্যাদি শিক্ষার সহায়কও হতে পারে পি. সি.। এছাড়া টিভির পর্দায় লেখা স্বরচিত গল্প, কবিতা, গান বা রচনা ছেলেমেয়েরা কী-বোর্ডের চাবি টিপে সঞ্চয় করে রাখতে পারে পি. সি.-র স্মৃতি ভাণ্ডারে।

একটু দামি পার্সোনাল কম্পিউটারে কম্পিউটার এইডেড ডিজাইন বা ক্যাড (CAD) করা সম্ভব। এতে সঠিক সফটওয়্যার-এর সাহায্যে স্থাপত্য, যন্ত্র-বিজ্ঞান ও অন্যান্য বৈজ্ঞানিক বা প্রযুক্তিগত কাজের সুন্দর সুন্দর নকশা তৈরি করা হয়। ক্যাড-এর ক্ষেত্রে প্রিণ্টারের পরিবর্তে প্লটার (চিত্রাঙ্কন করার যন্ত্র) সরাসরি ব্যবহার করা হয়।

ওয়ার্ড প্রসেসর

ওয়ার্ড প্রসেসিং—অর্থাৎ, সোজা কথায় কম্পিউটারে যাবতীয় চিঠি বা কোনও লেখার মুসাবিদা করা ও পরে প্রিণ্টারে তা ছাপিয়ে বের করা পি. সি. প্রযুক্তির আর এক অনবদ্য অবদান। টাইপ মেশিনের বদলে ওয়ার্ড প্রসেসর ব্যবহারের অনেক সুবিধা আছে। ওয়ার্ড প্রসেসিং-এর মজা হল কম্পিউটারের কী-বোর্ডের মাধ্যমে সমস্ত চিঠি বা লেখার মুসাবিদা টাইপ করে তা ভি. ডি. ইউ.-র পর্দায় দেখে ভুল সংশোধন করা খুব সহজ। লেখায় কোনও বানান শোধরাতে হলে, কোনও শব্দ বা কোনও লাইন পালটাতে হলে, বা নতুন কিছু জুড়তে হলে সে সবই ওয়ার্ড প্রসেসরের সাহায্যে অনায়াসে করে নেওয়া যায়। কোনও বানানের কোনও অক্ষর বাদ পড়লে কী-বোর্ডের বিশেষ বোতামের সাহায্যে ওই বানানের মধ্যে প্রয়োজনীয় জায়গায় শূন্যস্থান সৃষ্টি করা হয়। তখন বাদ পড়া অক্ষরটি ওই শূন্যস্থানে সহজেই লিখে দেওয়া যায়।

তাহলে আমরা দেখতে পাচ্ছি, ওয়ার্ড প্রসেসরে প্রধানত দুটো সুবিধে আছে ঃ

এক ঃ চিঠি বা কোনও লেখার সংশোধন ও পরিমার্জন।

দুই ঃ যা কিছু পর্দায় দেখা গেল তা কম্পিউটারের স্মৃতিভাণ্ডারে সঞ্চিত রইল ভবিষ্যতে বারবার ব্যবহারের অপেক্ষায়।

মোটের ওপর, ওয়ার্ড প্রসেসরে খুশি মতো যে কোনও লেখার জটিল সম্পাদনার কাজ অতি সহজে সেরে নেওয়া যায়। চিঠি অথবা পাণ্ডুলিপির পৃষ্ঠা যথাযথ, শুদ্ধ এবং মনের মতো হলে তবেই প্রিণ্টারে কাগজ দিয়ে প্রিণ্ট আউট চাবির মারফৎ তা ছাপানোর নির্দেশ দেওয়া হয়। তখন পর্দায় ফুটে ওঠা সমস্ত লেখা কাগজে ছাপা হতে থাকবে। উপযুক্ত প্রোগ্রামিং-এর সাহায্যে কাগজে ছাপা হওয়া লেখার মার্জিন, স্পেসিং, প্যারাগ্রাফ শুরু হওয়ার বিশেষ কোনও নির্দেশ ইত্যাদি নির্দিষ্ট করে দেওয়া যায়। তখন প্রিণ্টার সেই অনুযায়ী লেখাটি কাগজে ছেপে দেয়।

এ সব থেকেই পার্সোনাল কম্পিউটারের জনপ্রিয়তার কারণ অনুমান করা যায় না কি?

---

৩ □ কম্পিউটারের ভাষা

মানুষের ভাষা ও কম্পিউটারের ভাষা

প্রত্যেক প্রাণীরই নিজস্ব ভাষা আছে। পাখি কথা বলে পাখির ভাষায়। পশু ভাব-বিনিময় করে তার নির্দিষ্ট ভাষায়। আর মানুষ? মানুষ ভাব বিনিময় করতে পারে বিভিন্ন ভাষায়। সুতরাং ভাষা হচ্ছে পরস্পরের সঙ্গে ভাব বিনিময়ের প্রধান মাধ্যম।

কম্পিউটার একটি শক্তিশালী যন্ত্র। নানান জটিল কাজ সে করতে পারে। কিন্তু কী কাজ তাকে করতে হবে সেটা উপযুক্ত নির্দেশ দিয়ে তাকে বোঝাতে না পারলে দামী যন্ত্রটা স্রেফ বোঝা হয়ে বসে থাকবে। সুতরাং কম্পিউটারকে দিয়ে কাজ করাতে হলে তাকে এমন ভাষায় নির্দেশ দিতে হবে যা সে বুঝতে পারে এবং অক্ষরে অক্ষরে সেই সব নির্দেশ পালন করতে পারে। তবেই তার সঙ্গে 'ভাব-বিনিময়' করা সম্ভব। আর তখন কম্পিউটারও হয়ে উঠবে 'কাজের' যন্ত্র।

আগের পরিচ্ছেদে আমরা দেখেছি কম্পিউটার কাজ করে বাইনারি পদ্ধতিতে। সুতরাং যে ভাষা সে প্রকৃতই বুঝতে পারে তা হল বাইনারি ভাষা। এই ভাষা আমাদের ভাব-বিনিময়ের ভাষার (বাংলা, ইংরেজি বা হিন্দি) তুলনায় অনেক সহজ সরল। তবে কম্পিউটারের ভাষা ব্যবহার করতে হয় অত্যন্ত নিখুঁতভাবে। আমরা কথা বলার সময় বা লেখার সময় সামান্য ভুল করলেও ভাব-বিনিময়ে তেমন একটা অসুবিধে হয় না। কিন্তু কম্পিউটারের ভাষার ক্ষেত্রে এক চুল ভুল হলেই সে আর কাজ করবে না। কারণ তার মস্তিষ্ক বা সেণ্ট্রাল প্রসেসিং ইউনিট আমাদের মস্তিষ্কের মতো এমন ক্ষমতাবান নয় যে ভুল নির্দেশ পেলেও সেটার ভুল শুধরে নিয়ে সঠিক অর্থটি বুঝে নেবে। সুতরাং কম্পিউটারের ভাষা প্রয়োগ করার সময়ে সেই ভাষার 'ব্যাকরণ' কাঁটায় কাঁটায় মেনে চলতে হবে—এমন কি তার দাঁড়ি-কমা পর্যন্ত। তা না হলে কম্পিউটার নির্দেশ অনুযায়ী একটি কাজও করবে না।

কম্পিউটারের ভাষা প্রধানত দু রকমের ঃ

১। নিচুদরের ভাষা বা লো লেভেল ল্যাঙ্গুয়েজ।

২। উঁচুদরের ভাষা বা হাই লেভেল ল্যাঙ্গুয়েজ।

নিচুদরের ভাষা বা লো লেভেল ল্যাঙ্গুয়েজ

এই ভাষাকে সাধারণত দুটি শ্রেণীতে ভাগ করা যায়ঃ মেশিন ল্যাঙ্গুয়েজ ও অ্যাসেম্বলি ল্যাঙ্গুয়েজ।

মেশিন ল্যাঙ্গুয়েজ বা যান্ত্রিক ভাষা কম্পিউটারকে সরাসরি তার ভাষাতেই নির্দেশ দেয়—অর্থাৎ, বাইনারি ভাষা ব্যবহার করে। সন্দেহ নেই, এতে প্রোগ্রামার-এর কাজটি বেশ কঠিন হয়ে পড়ে। প্রথমত, তাঁকে অনভ্যস্ত একটি ভাষায় সমস্ত নির্দেশ লিখতে হয়। যেহেতু আমাদের পরিচিত ভাষার সঙ্গে বাইনারি ভাষার কোনও মিলই নেই, তাই প্রোগ্রামারকে অনেক বেশি সতর্ক হয়ে প্রোগ্রাম লিখতে হয়, আর এতে সময়ও লাগে অনেক বেশি। তাছাড়া কম্পিউটারকে কোনও কাজ করাতে যান্ত্রিক ভাষায় নির্দেশ দিতে চাইলে নির্দেশের সংখ্যাও লাগে বেশি। ধরা যাক, আমরা নিচের সমীকরণটি কম্পিউটারকে জানাতে চাই।

$$A=B\times C+D$$

অর্থাৎ, B, C ও D এর মান জানিয়ে দিলে কম্পিউটার অঙ্ক কষে A এর মান নির্ণয় করতে পারবে। যান্ত্রিক ভাষায় এই সমীকরণটি কম্পিউটারকে বোঝাতে গেলে প্রায় পাঁচটি নির্দেশের দরকার হবে। অথচ আমাদের প্রচলিত ভাষায় ওপরের এক লাইন নির্দেশই যথেষ্ট। অর্থাৎ, $A=B\times C+D$।

মেশিন ল্যাঙ্গুয়েজ বা যান্ত্রিক ভাষার এই অসুবিধে কিছুটা কমাতে অ্যাসেম্বলি ল্যাঙ্গুয়েজ-এর প্রচলন হয়েছিল। এই ভাষা ব্যবহার করে যান্ত্রিক ভাষার বাইনারি নির্দেশগুলো কিছুটা সংক্ষেপ করা সম্ভব হয়েছে। কিন্তু তা হলেও আমাদের প্রচলিত ভাষার সঙ্গে এর তেমন কোনও মিল নেই। ফলে, এই ভাষায় প্রোগ্রাম লেখার কাজটা অনেকটা বাইনারি ভাষার মতোই পরিশ্রমের ও সময়সাপেক্ষ। যেমন, উদাহরণ হিসেবে বলা যেতে পারে, বাইনারি ভাষার 1001 সংখ্যাটি হয়তো কম্পিউটারকে গুণ করার নির্দেশ দেয়। সুতরাং গুণ করার নির্দেশ দিতে হলে মেশিন ল্যাঙ্গুয়েজে প্রোগ্রামারকে 1001 লিখতে হবে। অ্যাসেম্বলি ল্যাঙ্গুয়েজে এই একই নির্দেশ হয়তো 'MULT' শব্দটির চেহারা নেবে। তখন কম্পিউটার সেটিকে অনুবাদ করে 1001 করে নেবে এবং বুঝতে পারবে যে তাকে গুণ করার নির্দেশ দেওয়া হয়েছে। এইভাবে কিছু কিছু সংক্ষিপ্ত সংকেত ও বাইনারি ভাষা ব্যবহার করে অ্যাসেম্বলি ল্যাঙ্গুয়েজে প্রোগ্রাম লেখা হয়।

কিন্তু স্পষ্টই বোঝা যাচ্ছে, যে-কোনও ধরনের নিচুদরের ভাষা ব্যবহার করে কম্পিউটারকে নির্দেশ দেওয়ার কাজটা যেমন সময় সাপেক্ষ তেমনই অসুবিধেজনক। এই দুটি বাধা কাটিয়ে ওঠার জন্যেই তৈরি হয়েছে কম্পিউটারের উঁচুদরের ভাষা বা হাই লেভেল ল্যাঙ্গুয়েজ।

উঁচুদরের ভাষা বা হাই লেভেল ল্যাঙ্গুয়েজ

1950-এর দশকের মাঝামাঝি উঁচুদরের ভাষার প্রচলন শুরু হয়। সাধারণত এই ভাষাগুলোকেই বলা হয় 'প্রোগ্রামিং ল্যাঙ্গুয়েজ'। উঁচুদরের ভাষায় প্রোগ্রাম লিখে ফেলা যায় অতি দ্রুত, আর তার সংস্কারও করা যায় বেশ সহজে। বেশির-ভাগ উঁচুদরের ভাষাই 'আমেরিকান ন্যাশনাল স্ট্যান্ডার্ড ইন্সটিটিউট'-এর প্রাথমিক নির্দেশ মেনে তৈরি করা।

উঁচুদরের ভাষায় কোনও কম্পিউটারকে যে-সব নির্দেশ দেওয়া হয় সেগুলো মেশিন ল্যাঙ্গুয়েজে অনুবাদ করে নেয় কম্পিউটার নিজেই। তা না হলে প্রোগ্রামারের দেওয়া নির্দেশ সে বুঝতে পারবে না। কম্পিউটারের ভেতরে যে অংশ এই অনুবাদের ব্যবস্থা করে তার নাম 'কম্‌পাইলার' অথবা 'ইন্টারপ্রিটার'। কোনও কোনও কম্পিউটারে 'কম্‌পাইলার' ব্যবহার করা হয়, আবার কোনও কম্পিউটারে থাকে 'ইন্টারপ্রিটার'।

উঁচুদরের ভাষার প্রধান সুবিধেগুলো হল ঃ

১। বিভিন্ন ধরনের কম্পিউটারের জন্য সামান্য পরিবর্তন করে অথবা কোনওরকম পরিবর্তন না করেই এই ভাষা ব্যবহার করা যায়।
২। মেশিন ল্যাঙ্গুয়েজ বা অ্যাসেম্বলি ল্যাঙ্গুয়েজের তুলনায় এই ভাষা শেখা সহজ।
৩। এই ভাষায় প্রোগ্রাম লিখতে সময় লাগে অনেক কম।

উঁচুদরের ভাষাকে মোটামুটিভাবে চারটি শ্রেণীতে ভাগ করা যায় ঃ

১। বিজ্ঞান সংক্রান্ত ভাষা, যেমন, FORTRAN (ফরট্রান), BASIC (বেসিক), ALGOL (অ্যালগল)।
২। ব্যবসা সংক্রান্ত ভাষা, যেমন, COBOL (কোবল)।
৩। বিশেষ কাজের জন্য প্রয়োজনীয় ভাষা, যেমন, CAL (ক্যাল), SNOBOL (স্নোবল)।
৪। বহুমুখী ব্যবহারিক ভাষা, যেমন, PL/1 (পি. এল. ওয়ান), PASCAL (পাস্কাল)।

কম্পিউটারের এতরকমের ভাষার মধ্যে চারটি ভাষা বেশ জনপ্রিয়। এগুলো হল ঃ ফরট্রান, কোবল, পাস্কাল ও বেসিক।

ফরট্রান ঃ কম্পিউটার জগতে প্রথম যে উঁচুদরের ভাষা চালু হয়েছিল তার নাম ফরট্রান (FORTRAN)। এর পুরো নাম হল FORmula TRANslator (ফর্মুলা ট্রানস্লেটর)। 1956 সালে IBM 704 কম্পিউটারে এই ভাষা ব্যবহার করা হয়। বিভিন্ন জটিল অঙ্কের সমাধান করার জন্যই ফরট্রান তৈরি করা

হয়েছিল। এই ভাষায় গাণিতিক সমস্যাকে সহজ ও স্বাভাবিক ইংরিজিতে প্রকাশ করে কম্পিউটারকে সমাধানের নির্দেশ দেওয়া যায়। পরে, ষাটের দশক থেকে আজ পর্যন্ত, ফরট্রান-এর বেশ কয়েকটি উন্নত রূপ প্রচলিত হয়েছে। যেমন. ফরট্রান-টু, ফরট্রান-ফোর, ফরট্রান-সেভেণ্টি সেভেন।

কোবল ঃ ব্যবসায়িক সমস্যা ইত্যাদি সমাধানের জন্য 'ইউ. এস. ডিপার্টমেণ্ট অফ ডিফেন্স' 1958 সাল নাগাদ একটি আন্তর্জাতিক উঁচুদরের ভাষা চালু করে। এরই নাম কোবল (COBOL), যার পুরো নাম হল COmmon Business Oriented Language (কমন বিজনেস ওরিয়েণ্টেড ল্যাঙ্গুয়েজ)। দৈনন্দিন অফিসের কাজে বা ব্যবসা সংক্রান্ত কাজে কোবল অপরিহার্য বলা যায়।

পাস্কাল ঃ পাস্কাল (PASCAL) ভাষার জন্ম হয় 1968 সালে। এটি আবিষ্কার করেন নিকলস ওয়ার্থ। ফরাসী বিজ্ঞানী ব্লাজ পাস্কালের নামে এই ভাষাটির নাম দেওয়া হয়। ছাত্রদের শিক্ষা দেবার কাজে পাস্কাল একটি উপযুক্ত উঁচুদরের ভাষা। এই ভাষার বৈশিষ্ট্য হল তার গঠন। নিয়ম মাফিক গঠনের জন্য এই ভাষায় প্রোগ্রাম করা ও বোঝা অনেক সুবিধেজনক। তাছাড়া এই ভাষা ব্যবহার করলে প্রোগ্রামার প্রোগ্রামিং-এর কাজে অনেক বেশি স্বাধীনতা পায়। এইসব কারণেই ভাষাটি বর্তমানে বেশ জনপ্রিয়।

বেসিক ঃ 1960-এর দশকের মাঝামাঝি বেসিক (BASIC) ভাষার প্রচলন হয়। এই ভাষাটির জনক আমেরিকার ডার্টমাউথ কলেজের দুই অধ্যাপক জন কিমেনি ও টমাস কুর্ৎস। কতকগুলো শব্দের আদ্যক্ষর নিয়ে সংক্ষিপ্ত BASIC নামটি তৈরি হয়েছে। এই ভাষার বিস্তারিত নামটি হল Beginners All-purpose Symbolic Instruction Code (বিগিনার্স অল-পারপাস সিম্বলিক ইন্সট্রাকশন কোড)। এমনিতে বেসিক কথার অর্থ বুনিয়াদী। কম্পিউটার প্রোগ্রামিং শেখার প্রথম ধাপ হিসেবে বুনিয়াদী ভাষা বা বেসিক-এর জুড়ি নেই। এই ভাষা শেখার জন্য প্রোগ্রামিং-এর জটিল কলাকৌশল শেখার প্রয়োজন হয় না। যাদের কম্পিউটার ব্যবহারের কোনও অভিজ্ঞতাই নেই বা কম্পিউটার প্রোগ্রাম লেখার অ-আ-ক-খ জানা নেই, তাদের পক্ষে বেসিকই সবচেয়ে উপযুক্ত ভাষা। কারণ বেসিক-এর নিয়ম কানুন অন্যান্য উঁচুদরের ভাষার তুলনায় কম এবং সহজ। কিন্তু সহজ হলে কি হবে, বেসিক-এর ক্ষমতা অন্যান্য ভাষার চেয়ে নেহাৎ কম নয়। বিজ্ঞান ও প্রযুক্তিবিদ্যার অঙ্ক, হিসেবপত্র বা ব্যবসা সংক্রান্ত কাজে বেসিক সমানভাবে ব্যবহার করা যায়। এত সব গুণের জন্যই বিভিন্ন উঁচুদরের ভাষার মধ্যে বেসিক-ই বর্তমানে সবচেয়ে জনপ্রিয়। আজকালকার ছোট কম্পিউটারগুলো প্রায় সর্বজনীনভাবে বেসিক ভাষা ব্যবহার করে। এছাড়া বিভিন্ন 'কম্পিউটার গেম্‌স্‌' তৈরিতেও বেসিক ভাষা কাজে লাগে। এইভাবে বেসিক-এর ব্যাপক ব্যবহার ক্রমশই আরও ছড়িয়ে পড়ছে কম্পিউটারের জগতে।

---

৪ □ বেসিক প্রোগ্রামিং : প্রথম পাঠ

বেসিক প্রোগ্রামের গঠন

বেসিক প্রোগ্রামের মাধ্যমে কম্পিউটারকে যে-সব নির্দেশ দেওয়া হয় তার প্রত্যেকটি নির্দেশকে এক একটি ভিন্ন বক্তব্য বা 'স্টেটমেণ্ট'-এর আকারে প্রকাশ করা হয়। বেশ কিছু সংখ্যক বক্তব্যকে পরপর সাজিয়ে তৈরি হয় একটি সম্পূর্ণ বেসিক প্রোগ্রাম। স্টেটমেণ্ট বা বক্তব্যগুলো যেভাবে সাজানো থাকে কম্পিউটার ঠিক সেই ক্রমানুসারে সেগুলো পড়ে নেয় এবং সাধারণত একই ক্রমানুসারে নির্দেশগুলো পালন করতে থাকে।

সঠিক বেসিক প্রোগ্রাম লিখতে গেলে নিচের নিয়মগুলো মেনে চলতে হবে। অবশ্য বেসিক-এর কয়েকটি উন্নত সংস্করণে এই নিয়মের সামান্য হেরফের লক্ষ্য করা যেতে পারে।

বেসিক ভাষার মূল নিয়মগুলো হল :

১। প্রত্যেকটি নির্দেশকে একটি ভিন্ন লাইনে লিখতে হবে। এক্ষেত্রে সাধারণত একটি লাইনের অর্থ মোট 80টি অক্ষর বা চিহ্ন। অবশ্য কোনও কোনও কম্পিউটারে 132টি পর্যন্ত অক্ষর বা চিহ্ন একই লাইনে লেখা যায়।

২। একটি নির্দেশ কখনও এক লাইনের চেয়ে বড় হবে না। অর্থাৎ, কোনও নির্দেশ এক লাইনে শেষ না হলে তাকে পরের লাইনে টেনে নিয়ে শেষ করলে চলবে না। ফলে নির্দেশগুলো এমনভাবে ঠিক করতে হবে যাতে সেগুলো এক একটি লাইনে শেষ হয়।

৩। প্রত্যেকটি নির্দেশের শুরুতে একটি ধনাত্মক পূর্ণ সংখ্যা থাকতে হবে। এই সংখ্যা নির্দেশের নম্বর বা লাইনের নম্বর সূচিত করে—যাকে ইংরিজিতে বলা হয় 'স্টেটমেণ্ট নাম্বার' বা 'লাইন নাম্বার'। কোনও দুটি নির্দেশের সূচক সংখ্যা কখনও সমান হবে না।

৪। নির্দেশগুলো যখন ধারাবাহিক ভাবে লেখা হবে তখন তাদের নির্দেশ সূচক সংখ্যাগুলো ক্রমশ বাড়তে থাকবে। পরে লেখা নির্দেশের সূচক সংখ্যা কখনও আগে লেখা নির্দেশের সূচক সংখ্যার চেয়ে ছোট হবে না।

৫। প্রতিটি নির্দেশ-সূচক সংখ্যার ঠিক পরেই বেসিক ভাষার কোনও একটি 'কী-ওয়ার্ড" বা মূল শব্দ ব্যবহার করতে হবে। কম্পিউটারকে ঠিক কোন্ ধরনের নির্দেশ পালন করতে হবে এই মূল শব্দটি তার ইঙ্গিত দেয়।

৬। যে কোনও নির্দেশের মাঝে খুশি মতো 'ব্ল্যাঙ্ক স্পেস' বা শূন্য স্থান রাখা যেতে পারে। সুবিধে মতো খালি জায়গা রাখলে বড় প্রোগ্রাম পড়তে কোনও অসুবিধে হয় না।

বেসিক প্রোগ্রামে নির্দেশবিহীন খালি লাইনও রাখা যায়। এর জন্য লাইন নম্বর বসিয়ে তার পরেই শূন্যস্থান বা ব্ল্যাঙ্ক স্পেস নির্দেশক বোতাম টিপতে হয় (তবে বেসিকের কোনও কোনও সংস্করণে এরকম খালি লাইন রাখা যায় না)। আর সব শেষে টিপতে হয় কী-বোর্ড-এর 'RETURN' নামক বোতামটি। এখানে মনে রাখতে হবে, বেসিক প্রোগ্রামের প্রতিটি নির্দেশ লেখার পরে পরেই 'রিটান" বোতামটি টিপতে হবে সব সময়। কারণ এই বোতামটি টিপলে তবেই কোনও নির্দেশ কম্পিউটারের 'মগজে' প্রবেশ করে। এছাড়া, কম্পিউটারকে প্রোগ্রাম অনুযায়ী কাজ করার চূড়ান্ত নির্দেশ দিতেও 'রিটান" বোতামটি টিপতে হয়। একটু পরেই এই বোতাম নিয়ে বিশদ আলোচনা করা যাবে।

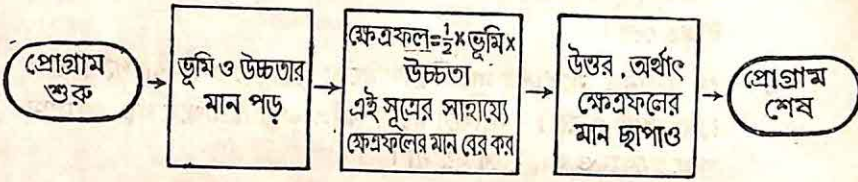
যদি কোনও লাইন নম্বর বসানোর পরেই আর কোনও বোতাম না টিপে সরাসরি 'রিটান" বোতামটি টেপা হয় তাহলে সেই লাইনটি নির্দেশ তালিকা থেকে তৎক্ষণাৎ বর্জিত হবে।

একটি উদাহরণের সাহায্যে ওপরের নিয়মগুলোর প্রয়োগ ব্যাখ্যা করা যাক। মনে করি, কোনও ত্রিভুজের ক্ষেত্রফল নির্ণয় করতে হবে। এই কাজের জন্য আমাদের দরকার ত্রিভুজটির ভূমি ও উচ্চতা। তাহলেই আমরা হিসেব করে বের করতে পারব ত্রিভুজের ক্ষেত্রফল। এই কাজের জন্য আমরা নিচের সূত্রটির সাহায্য নেব ঃ

$$\text{ত্রিভুজের ক্ষেত্রফল} = \frac{1}{2} \times \text{ভূমি} \times \text{উচ্চতা}$$

কম্পিউটারে যখন এই অঙ্কটি কষা হবে তখন কম্পিউটার কয়েকটি ধাপে কাজটি সম্পন্ন করবে। যেমন, প্রথমেই তাকে জানাতে হবে ত্রিভুজের ভূমি ও উচ্চতার মান। তারপর ওপরের সূত্রটির সাহায্য নিয়ে সে বের করবে ত্রিভুজের ক্ষেত্রফল। পরবর্তী ধাপে কম্পিউটার ক্ষেত্রফলের মান প্রিণ্টারের কাগজে ছাপিয়ে দেবে অথবা ভিসুয়াল ডিসপ্লে ইউনিট বা টিভি-র পর্দায় তা ফুটিয়ে তুলবে। তারপরই শেষ হবে তার কাজ। পরের পৃষ্ঠার ছবিটির

সাহায্যে কম্পিউটারের কাজের বিভিন্ন ধাপগুলো বেশ সহজে দেখানো যেতে পারে।



এবারে এই ধাপগুলো একটি বেসিক প্রোগ্রামের সাহায্যে প্রকাশ করা যাক :-

[1]

```
10 INPUT B, H  RETURN
20 LET A=(1/2) * B * H  RETURN
30 PRINT A  RETURN
40 END  RETURN
```

এই প্রোগ্রামে B-র অর্থ Base বা ভূমি, H-এর অর্থ Height বা উচ্চতা, আর A-র অর্থ Area বা ক্ষেত্রফল।

দেখা যাচ্ছে যে এই প্রোগ্রামটিতে মোট চারটি নির্দেশ ব্যবহার করা হয়েছে। এই চারটি নির্দেশই ভিন্ন ভিন্ন লাইনে লেখা হয়েছে। প্রত্যেক লাইনের নির্দিষ্ট একটি লাইন নম্বর রয়েছে। লাইন নম্বরগুলো শুরু থেকে শেষ পর্যন্ত ক্রমশ সংখ্যায় বেড়েছে (যেমন, 10, 20, ··· 40)। আর লাইনগুলোতে বেসিক ভাষার কী-ওয়ার্ড বা মূল শব্দ INPUT, LET, PRINT ও END ব্যবহার করা হয়েছে। এই নির্দেশগুলো সবই কী-বোর্ডের বোতাম টিপে কম্পিউটারের ভি. ডি. ইউ.-তে ফুটিয়ে তুলতে হবে এবং প্রতিটি নির্দেশের শেষে নিয়মমাফিক রিটার্ন বোতামটি টিপতে হবে।

প্রোগ্রামের প্রথম লাইনে কম্পিউটার ইনপুট তথ্য B ও H-এর মান পড়বে। দ্বিতীয় নির্দেশ অনুযায়ী সে A-র মান নির্ণয় করবে। তৃতীয় নির্দেশ A-র মানকে ভি. ডি. ইউ.-তে ফুটিয়ে তুলবে বা 'ছাপিয়ে' দেবে। আর শেষ নির্দেশটি কম্পিউটারকে বুঝিয়ে দেবে যে প্রোগ্রামটি শেষ হয়েছে।

প্রোগ্রামটি কম্পিউটারকে দিয়ে পালন করাতে গেলে প্রথমেই বোতাম টিপে RUN কথাটি লিখে রিটার্ন বোতাম টিপতে হবে। তখন একটি জিজ্ঞাসা চিহ্ন (?) ফুটে উঠবে টিভি পর্দায়। অর্থাৎ, কম্পিউটার B ও H-এর মান জানতে চাইছে। তখন B ও H-এর সাংখ্যমান উপযুক্ত সংখ্যার বোতাম

টিপে জানাতে হবে। তারপর রিটার্ন বোতামটি টিপলেই পাওয়া যাবে A-র মান। এই প্রোগ্রামে গুণ চিহ্ন হিসেবে '*' চিহ্নটি ব্যবহার করা হয়েছে আর ভাগের জন্য '/' চিহ্নটি। কম্পিউটারের ক্ষেত্রে এই চিহ্নগুলোই প্রচলিত।

বেসিক ভাষার কয়েকটি সুবিধেজনক দিক হল ঃ

১। বেসিক খুব সহজে শেখা যায় আর ব্যবহার করতেও বেশ মজা। বেসিক ভাষা অনেকটা প্রোগ্রামার এর বন্ধুর মতো।

২। বড়-ছোট সব ধরনের কম্পিউটারেই বেসিক ব্যবহার করার সুযোগ রয়েছে। বেশির ভাগ পি. সি.-তেই প্রোগ্রামিং-এর অন্যতম ভাষা বেসিক।

৩। বেসিক ভাষা সহজ হলেও কম শক্তিশালী নয়। বেসিকের সাহায্যে সহজ-কঠিন নানান রকমের গাণিতিক বা অন্যান্য সমস্যার সমাধান করা যায়।

৪। বেসিকের বিভিন্ন রকমের সংস্করণ থাকলেও তাদের মূল গঠন বা বৈশিষ্ট্যের তেমন কোনও তফাৎ নেই। ফলে এক কম্পিউটারের উপযোগী বেসিক প্রোগ্রাম অন্য আর একটি কম্পিউটারে সহজেই ব্যবহার করা যায়। এর জন্য বেশিরভাগ ক্ষেত্রেই কোনও পরিবর্তন বা পরিমার্জনের প্রয়োজন পড়ে না।

৫। প্রথম কম্পিউটার প্রোগ্রাম শেখার জন্য বেসিক ভাষাই হল আদর্শ। বেসিক ভাষা একবার রপ্ত হলে কম্পিউটারের অন্যান্য ভাষা শেখার কাজটা অনেক সহজ হয়ে যায়।

প্রোগ্রামিং-এ হাতে-খড়ি

প্রোগ্রামিং শুরু করার শুরুতেই একটা কথা বলা দরকার। এই অধ্যায় ও তার পরবর্তী পঞ্চম অধ্যায়ে প্রোগ্রামিং সম্পর্কে যে-সব আলোচনা করা হবে তার প্রায় সবটুকুই কম্পিউটার নিয়ে অভ্যাস করা প্রয়োজন। আর এই অভ্যাস করতে গেলে সপ্তম অধ্যায়ের সাহায্য নিতে হবে। সেই কারণেই পাঠককে চতুর্থ ও পঞ্চম অধ্যায়ের সঙ্গে সঙ্গে প্রয়োজন মতো সপ্তম অধ্যায়ের সাহায্য নিতে অনুরোধ করছি।

আমাদের প্রথম প্রোগ্রামটি থেকেই একটা জিনিস লক্ষ্য করা যায়—সেটা হল, প্রোগ্রামিং-এর সময়ে বিভিন্ন শব্দকে সংক্ষেপে এক বা একাধিক অক্ষরের সাহায্যে প্রকাশ করা হয়। যেমন, প্রথম উদাহরণের B, H ও A যথাক্রমে ভূমি, উচ্চতা ও ক্ষেত্রফলের সংক্ষিপ্ত সংকেত।

ধরা যাক, কোনও ছাত্রের স্কুল ব্যাগে পাঁচটা বই, দুটো পেন্সিল আর চারটে খাতা আছে। এই তথ্যগুলো সংক্ষেপে প্রকাশ করা যায় কেমন করে? মনে করি, T=ছাত্রটির ব্যাগে যে-সব জিনিস আছে, B=বই, P=পেন্সিল, এবং K=খাতা। তাহলে ছাত্রটির ব্যাগে কী কী জিনিস আছে সেটা আমরা একটা ফর্মুলা বা সূত্রের সাহায্যে সংক্ষেপে প্রকাশ করতে পারি। অর্থাৎ,

$$T=5 * B+2 * P+4 * K$$

এবারে বেসিক প্রোগ্রামের সাহায্যে একটা অঙ্কের উত্তর কষে ফেলা যাক।

রামের বয়েস 10 বছর এবং শ্যামের বয়েস 8 বছর হলে উভয়ের মিলিত বয়েস কত?

মনে করি, রামের বয়েস=A, শ্যামের বয়েস=B, আর উভয়ের মোট বয়েস=C, তাহলে সহজ বেসিক প্রোগ্রামটি হবে এই রকম ঃ

2

```
10  INPUT A, B
20  LET C=A+B
30  PRINT C
40  END
```

আগের প্রোগ্রামের মতো এখানেও কম্পিউটার A ও B-এর মান জানতে চাইবে। উপযুক্ত বোতাম টিপে সেই মানগুলো জানালেই কম্পিউটার C বা বয়েসের যোগফল জানিয়ে দেবে। কম্পিউটারে প্রোগ্রাম করে অঙ্কটি করার সুবিধে এই যে রাম ও শ্যামের বয়েস যা-ই হোক না কেন, কম্পিউটার সহজেই তাদের বয়েসের যোগফল জানিয়ে দেবে। অথচ হাতে করে অঙ্কটি কষতে হলে রাম ও শ্যামের বিভিন্ন রকম বয়েসের জন্য বার বার যোগ করে উত্তর বের করতে হবে। সুতরাং কম্পিউটার আমাদের বহু সময় ও অনর্থক পরিশ্রম বাঁচিয়ে দেয়।

ওপরের বেসিক প্রোগ্রামটি একটু অন্য রকম করেও লেখা যায়। যেমন ঃ

3

```
10  READ A, B
20  LET C=A+B
30  PRINT C
40  DATA 10, 8
50  END
```

লক্ষ্য করলেই দেখা যাবে যে দুটি প্রোগ্রামের মধ্যে তফাৎ 10 নম্বর স্টেটমেন্ট-এ, আর শেষ প্রোগ্রামটিতে 40 নম্বর লাইনটি বাড়তি লেখা হয়েছে। এর কারণ, বেসিক ভাষার মূল শব্দ INPUT ও READ-এর অর্থের পার্থক্য। যদি প্রোগ্রামে INPUT স্টেটমেন্ট (প্রথম প্রোগ্রামের 10 নম্বর

স্টেটমেণ্ট) ব্যবহার করা হয় তাহলে প্রোগ্রাম করা শেষ হলে কী-বোর্ডের বোতাম টিপে লিখতে হবে

R U N

এবং তারপর রিটার্ন বোতামটি টিপলেই ফুটে উঠবে জিজ্ঞাসা চিহ্ন (অথবা ওই জাতীয় অন্য কোনও চিহ্ন)। তখন INPUT তথ্যগুলোর (যেমন A ও B) মান কী-বোর্ডের বোতাম টিপে প্রবেশ করাতে হবে কম্পিউটারে। মনে রাখতে হবে, প্রত্যেকটি তথ্য বোতাম টিপে লেখার পর রিটার্ন বোতামটি অবশ্যই টিপতে হবে। যেমন A ও B-র মান যদি যথাক্রমে 10 ও 8 হয় তাহলে 2 নম্বর প্রোগ্রামটি চালাতে হবে এইভাবে ঃ

```
10  INPUT A, B  RET   ( RET হল RETURN-এর সংক্ষিপ্ত রূপ )
20  LET C=A+B  RET
30  PRINT C  RET
40  END  RET
R U N         RET
? 1 0           RET
? 8               RET
18
```

অঙ্কের উত্তর 18 সংখ্যাটি শেষ রিটার্ন বোতামটি টেপার পরেই ফুটে উঠবে টিভি পর্দায়।

এখানে লক্ষ্য করার বিষয় ? চিহ্নটি। এই চিহ্ন পর্দায় ফুটে ওঠার অর্থ হল কম্পিউটার কোনও না কোনও ইনপুট তথ্যের মান জানতে চাইছে। এক্ষেত্রে কম্পিউটার A ও B-র মান জানতে চেয়েছে।

কিন্তু 3 নম্বর প্রোগ্রামের মতো READ নির্দেশ ব্যবহার করে প্রোগ্রাম করলে সেই প্রোগ্রাম RUN করানোর পরই উত্তর ফুটে উঠবে টিভি পর্দায়।

উদাহরণ হিসেবে আরও একটা সমস্যার সমাধান করে ফেলা যাক ঃ

x এর মান দেওয়া থাকলে ($1+x+2x^2$)-এর মান কত ?

এই সমস্যার বেসিক প্রোগ্রামটি হল ঃ

4

```
10 INPUT X
20 LET Y=1+X+2*(X↑2)
```

```
30 PRINT Y
40 END
```

এখানে x^2 বোঝাতে X ↑ 2 লেখা হয়েছে। বেসিক ভাষায় পাওয়ার বা ঘাত বোঝাতে '∧' বা '↑' চিহ্ন ব্যবহার করা হয়। যেমন,

$$x^2 \equiv X \wedge 2$$

অথবা, $x^2 \equiv X \uparrow 2$

এই প্রোগ্রামের বেলাতেও কম্পিউটার ? চিহ্নটি ফুটিয়ে তুলে X-এর মান জানতে চাইবে। তারপর X-এর মান দেওয়া মাত্রই সে ফুটিয়ে তুলবে Y-এর মান।

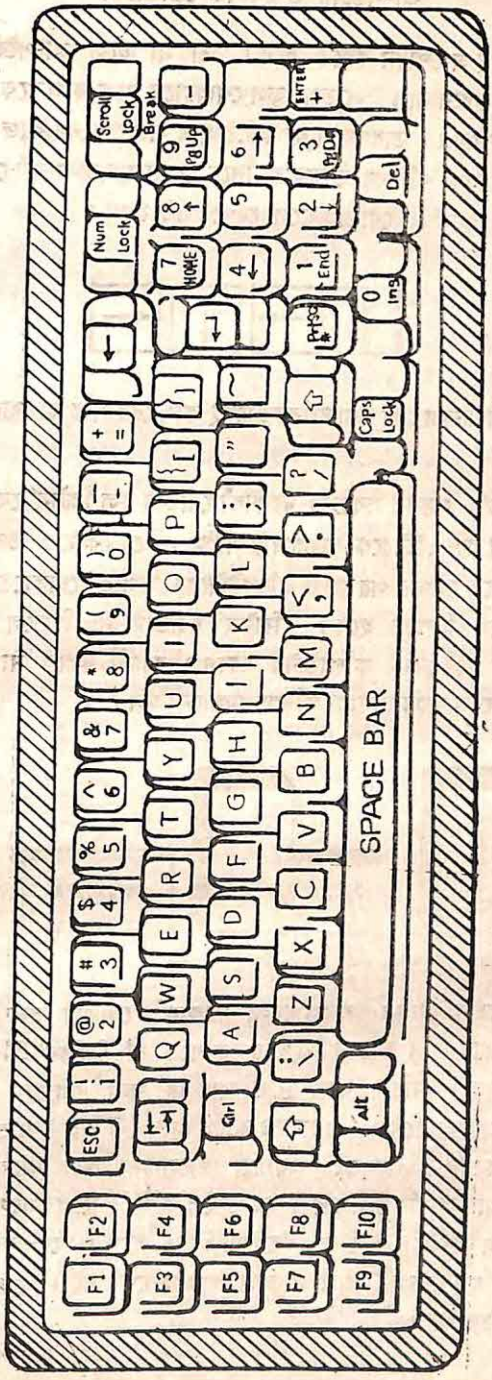
কী-বোর্ডের দরকারী বোতামের সঙ্গে পরিচয়

বেসিক প্রোগ্রামে হাতে খড়ি সেরে নেবার পর কী-বোর্ডটিকে আর একটু খুঁটিয়ে লক্ষ্য করা যাক।

দ্বিতীয় পরিচ্ছেদে পার্সোনাল কম্পিউটারের প্রাথমিক পরিচয় আমরা পেয়েছি। সেখানে দেখেছি, কম্পিউটারের একটা কী-বোর্ড থাকে। সাধারণ টাইপ মেশিনের কী-বোর্ডের সঙ্গে এই কী-বোর্ডের যথেষ্ট মিল রয়েছে। যেমন, বিভিন্ন সংখ্যা বা ইংরিজি অক্ষরমালার বোতামগুলো ঠিক একইভাবে সাজানো। তবে নতুন কিছু বোতামও কম্পিউটারের কী-বোর্ডে থাকে যা টাইপরাইটারের কী-বোর্ডে থাকে না।

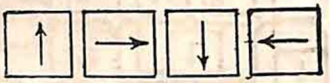
কম্পিউটারের বোতাম টিপে যখন আমরা 'টাইপ' করি তখন লেখাগুলো ফুটে ওঠে ভি. ডি. ইউ.-র পর্দায়। সুতরাং কোনও বোতাম টিপলে সেই চিহ্নটি পর্দার ঠিক কোন্ জায়গায় ফুটে উঠবে সেটা নিশ্চিতভাবে আমাদের জানা দরকার। কম্পিউটারের ভি. ডি. ইউ.-তে সেই কাজটি করে 'কার্সর' (CURSOR) নামের একটি চিহ্ন। চিহ্নটির চেহারা কম্পিউটার বিশেষে ভিন্ন ভিন্ন হলেও তার কাজের কোনও তফাৎ নেই। কোনও কম্পিউটারে এটিকে দেখতে ছোট আয়তক্ষেত্রের মতো, আবার কোথাও বা দুটি সমান্তরাল দাগ। দৃষ্টি আকর্ষণের জন্য টিভির পর্দায় কার্সর চিহ্ন সর্বদা দপদপ করে জ্বলে-নেভে। কী-বোর্ডের কোনও অক্ষরের বোতাম টিপলেই কার্সর-এর জায়গায় সেই অক্ষরটি ফুটে উঠবে, আর কার্সরটি চকিতে ডানপাশে এক ঘর সরে যাবে। অর্থাৎ, এরপর নতুন কোনও অক্ষরের বোতাম টিপলে সেই অক্ষরটি ফুটে উঠবে কার্সর-এর নতুন জায়গায় এবং কার্সর ডানদিকে আরও এক ঘর সরে গিয়ে দপদপ করে জ্বলতে থাকবে।

কোনও প্রোগ্রাম কম্পিউটারে লেখার সময় ভুল হতে পারে—তা সেই ভুল টাইপ করতে গিয়েই হোক বা নির্দেশ দেওয়ার গোলমালের জন্যই হোক। তখন



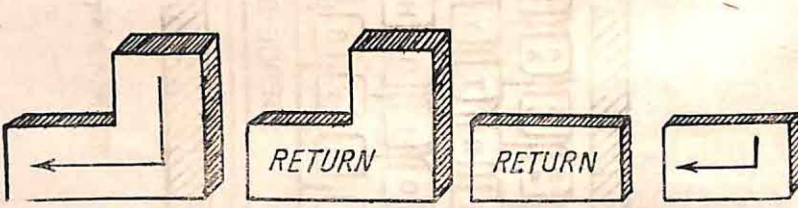
পার্সোনাল কম্পিউটারের কী-বোর্ড

সেই ভুল অবশ্যই সংশোধন করতে হবে। তা না হলে কম্পিউটার প্রোগ্রাম অনুযায়ী কাজ করবে না। সুতরাং ভুল শোধরাতে হলে কার্সরকে নিয়ে যেতে হবে ভুলের জায়গায়। তারপর শুরু করতে হবে সংশোধনের কাজ। কার্সরকে খুশি মতো ডানদিক, বাঁদিক, উপরে বা নিচে সরানোর জন্য কী-বোর্ডে চারটি বিশেষ বোতাম থাকে। বোতামগুলো দেখতে এই রকম ঃ



ছবির তীরচিহ্নগুলো স্বাভাবিকভাবেই কার্সরের সরে যাওয়ার দিক নির্দেশ করছে।

কী-বোর্ডের সম্ভবত সবচেয়ে দরকারী বোতাম হল রিটার্ন বোতামটি। এই বোতামটি যেন কম্পিউটারকে চালানোর সহিস। যে-কোনও তথ্য বা নির্দেশ ভি. ডি. ইউ.-তে লেখার পর সেটি কম্পিউটারের 'মগজে' ঢোকাতে গেলে অবশ্যই রিটার্ন বোতাম টিপতে হবে। রিটার্ন বোতাম না টিপলে যতই নির্দেশ দেওয়া হোক না কেন, কম্পিউটার কোনও কাজই করবে না। এই বিশেষ বোতামটির কয়েক রকম চেহারা ছবিতে দেখানো হল ঃ



কী-বোর্ডের আরও কতকগুলো দরকারী বোতাম হল ফাংশান-কী বা ফাংশান বোতাম। (আগের পৃষ্ঠার ছবিতে বাঁ দিকের F1, F2…ইত্যাদি দশটি বোতাম) বিভিন্ন কাজ দ্রুত সারবার জন্য এগুলো প্রয়োজন মতো ব্যবহার করা হয়ে থাকে। এছাড়াও আছে ট্যাব (TAB) ও অন্যান্য বোতাম। এগুলোর কয়েকটি পরবর্তী অধ্যায়ে আলোচনা করা যাবে। আর বাকী বোতামগুলোর কার্যকলাপ রপ্ত করতে গেলে কম্পিউটারের সঙ্গে যে হ্যান্ডবুক বা ম্যানুয়াল দেওয়া হয় সেটির সাহায্য নিতে হবে। যে-কোনও কোম্পানির কম্পিউটার ম্যানুয়ালটি হল অত্যন্ত গুরুত্বপূর্ণ বই। কোনও মতেই এটি হারানো উচিত নয়।

কম্পিউটারে প্রোগ্রাম শুরু

সাধারণত একটি কম্পিউটার বেসিক প্রোগ্রাম চালানো ছাড়াও অন্যান্য কাজ করতে পারে। সুতরাং কম্পিউটারের সুইচ অন করে প্রথমেই তাকে বোঝাতে হবে যে আমরা এখন বেসিক প্রোগ্রাম চালাতে চাই। এই বোঝানোর কায়দাটা বিভিন্ন ধরনের কম্পিউটারে বিভিন্ন। সেটা কম্পিউটার ম্যানুয়ালে দেওয়া থাকে। এখানে আমরা উদাহরণ হিসেবে সিনক্লেয়ার কম্পিউটারের পদ্ধতিটি জানালাম। সপ্তম অধ্যায়ে এ-বিষয়ে বিস্তারিত আলোচনা করা হয়েছে।

এই কম্পিউটারে সুইচ অন করার পর কী-বোর্ডে বোতাম টিপে লিখতে হবে 'RUN BASIC' কথা দুটি। তখন কম্পিউটার ভি. ডি. ইউ.-র মাধ্যমে জানতে চাইবে 'NEW OR OLD', অর্থাৎ আমরা নতুন কোনও প্রোগ্রাম করতে চাই, না পুরোনো কোনও প্রোগ্রামকে কম্পিউটারের মেমোরি বা স্মৃতি থেকে ডেকে আনতে চাই ভি. ডি. ইউ.-তে। যেহেতু আমরা নতুন প্রোগ্রাম করতে চাই সেহেতু আমরা তখন কী-বোর্ডে টাইপ করব 'NEW' শব্দটি। এবারে কম্পিউটার জানতে চাইবে 'NEW PROGRAM NAME', অর্থাৎ নতুন প্রোগ্রামের নাম কি ? আমাদের প্রত্যেকের যেমন এক একটা নাম আছে তেমনি কম্পিউটারের প্রতিটি প্রোগ্রামেরই এক একটা নাম থাকা দরকার। কেন না পরে দরকার পড়লে সেই নাম ধরে প্রোগ্রামটিকে আমরা স্মৃতি থেকে ডেকে নিতে পারব ভি. ডি. ইউ.-র পর্দায়। ধরা যাক, প্রোগ্রামটির নাম দেওয়া হল SUMIT 10, অর্থাৎ সুমিত নামের একটি ছেলে তার দশ নম্বর প্রোগ্রামটি করছে। একটা কথা এখানে মনে রাখা দরকার। যে-কোনও প্রোগ্রামের নামের প্রথম অক্ষর সব সময় কোনও বর্ণ (অর্থাৎ A, B, C··· ইত্যাদি) হতে হবে, কোনও সংখ্যা হলে (অর্থাৎ, 1, 2, 3 ···ইত্যাদি) ভুল হবে।

সুমিত কম্পিউটারে যে অঙ্কটির সমাধান করতে চায় মনে করি সেটা এইরকম ঃ রাম, শ্যাম ও যদু প্রত্যেকের কাছে যথাক্রমে পাঁচ টাকা, আট টাকা এবং তিন টাকা আছে। মধুর কাছে যদুর টাকার চেয়ে রাম ও শ্যামের টাকার গুণফলের অর্ধেক টাকা বেশি আছে। মধুর কাছে কত টাকা আছে ?

ধরে নেওয়া যাক, রামের টাকা = A, শ্যামের টাকা = B, যদুর টাকা = C এবং মধুর টাকা = X।

সুমিত নতুন প্রোগ্রামের নাম ভি. ডি. ইউ.-তে লেখামাত্রই কম্পিউটার লিখবে READY, অর্থাৎ, সুমিত এবারে তার প্রোগ্রামটি কী-বোর্ডে টাইপ করতে পারে। প্রোগ্রামের প্রতিটি লাইন বা নির্দেশ টাইপ করার পর রিটার্ন বোতামটি অবশ্যই টিপতে হবে।

পরের পৃষ্ঠায় সুমিতের প্রোগ্রামটি প্রথম থেকে লিখে দেখানো হল। যে-কথাগুলো কম্পিউটার নিজেই লিখবে সেগুলোর তলায় লাইন টেনে দেখানো হয়েছে।

```
5     RUN BASIC                               RET
        NEW OR OLD        NEW                 RET
      NEW PROGRAM NAME  SUMIT 10              RET
      READY
      10  READ A, B, C                        RET
      20  LET X=A*B / 2+C                     RET
      30  PRINT X                             RET
      40  DATA 5, 8, 3                        RET
      50  END                                 RET
      RUN                                     RET
      SUMIT 10     15-FEB-89    14 : 32 : 44
      X=23
      END AT 50
      READY
      SAVE SUMIT 10                           RET
      READY
      SCR ( অথবা KILL )                       RET
```

ওপরের প্রোগ্রামে 50 নম্বর নির্দেশ শেষ হওয়ার পর RUN লিখলে কম্পিউটার প্রোগ্রামটা চালাবে, অর্থাৎ উত্তর বের করবে। তারপর প্রোগ্রামের নাম, তারিখ ও সময় লেখার পর অঙ্কের উত্তরটা লিখবে। এরপর READY কথাটি লিখে কম্পিউটার বলতে চাইছে আর নতুন কোনও নির্দেশ দেবার আছে কিনা। ধরা যাক, সুমিতের এখন নতুন আর কোনও প্রোগ্রাম করার দরকার নেই। সুতরাং ও লিখল SAVE SUMIT 10। এই নির্দেশ পেলে কম্পিউটার সুমিতের প্রোগ্রামটা তার ওয়ার্কিং মেমোরি বা নিজস্ব স্মৃতি থেকে ফ্লপি ডিস্ক বা টেপ-এ সংরক্ষিত করে রাখবে। এর ফলে সুমিত এই প্রোগ্রামটাকে ভবিষ্যতে খুশি মতো ব্যবহার করতে পারবে। প্রোগ্রাম সংরক্ষণের পর আবার READY লেখা হলে SCR (অর্থাৎ SCRATCH) বা KILL লিখতে হবে—যার অর্থ আপাতত প্রোগ্রাম লেখার কাজ শেষ।

এবারে পাটিগণিতের সরল অঙ্কের একটি উদাহরণ দেওয়া যাক। পাটিগণিতের

সরল অঙ্কে আমরা তিন রকমের বন্ধনীর কথা জানি। প্রথম বন্ধনী ( ), দ্বিতীয় বন্ধনী { }, এবং তৃতীয় বন্ধনী [ ]। কিন্তু বেসিক ভাষায় শুধুমাত্র প্রথম বন্ধনীই আছে, দ্বিতীয় বা তৃতীয় বন্ধনী নেই। যেমন, একটা সরল অঙ্ক যদি এই রকম হয় ঃ

$$[2+3\{6-2(4\div 2)+1\}]$$

তাহলে বেসিক ভাষায় এর মান নির্ণয় করতে গেলে সরল অঙ্কটির চেহারা দাঁড়াবে একটু অন্যরকম ঃ

| 6 |

```
10  LET X=(2+3*(6-2*(4/2)+1))
20  PRINT X
30  END
```

এইবার প্রোগ্রামটি RUN করালেই কম্পিউটার X-এর মান, অর্থাৎ সরল অঙ্কটির উত্তর জানিয়ে দেবে। এখানে দেখা যাচ্ছে, বেসিক ভাষায় দ্বিতীয় বা তৃতীয় বন্ধনীর ক্ষেত্রেও প্রথম বন্ধনী ব্যবহার করা হয়। এই প্রসঙ্গে একটা জিনিস উল্লেখ করা প্রয়োজন। ওপরের প্রোগ্রামটিতে প্রতিটি নির্দেশের শেষে রিটার্ন বোতাম টেপার কথা বলা হয়নি। প্রতিটি নির্দেশের শেষে যে রিটার্ন বোতাম অবশ্যই টিপতে হবে সেকথা আগে বলা হয়েছে। তাই এই প্রোগ্রাম এবং পরবর্তী অন্যান্য প্রোগ্রামে রিটার্ন বোতাম টেপার কথা আমরা আলাদা করে আর উল্লেখ করব না।

এ-পর্যন্ত বেসিক ভাষায় অঙ্ক কষার পাঁচটি চিহ্ন আমরা দেখেছি—যোগ, বিয়োগ, গুণ, ভাগ এবং ঘাতসূচক চিহ্ন।

ঘাতসূচক সোজা তীর চিহ্ন বা টুপির মতো চিহ্নটি (পরিভাষায় যার নাম আপার ক্যারেট) সমানভাবে বিভিন্ন কম্পিউটারে ব্যবহার করা হয়। তবে আমরা ↑ চিহ্নটিকেই বরাবর ব্যবহারের জন্য বেছে নেব।

সুতরাং বেসিক ভাষায় +, −, *, / এবং ↑ চিহ্ন পাঁচটি হল ম্যাথামেটিক্যাল অপারেটর বা গাণিতিক সংকারক। যে-কোনও গাণিতিক সূত্র বা সমীকরণ লিখতে অপারেটর সংকেতগুলো ভীষণ জরুরী। বেসিক ভাষায় প্রোগ্রাম লিখতে গেলে প্রায়ই এই চিহ্নগুলোর দরকার পড়ে।

বেসিক প্রোগ্রামিং-এ হাতে-খড়ি যখন হয়েই গেল তখন কয়েকটা বেসিক প্রোগ্রাম লিখে আমরা পাঠককে তার অর্থ লিখতে দিলাম। এই প্রোগ্রামগুলো ঠিকমতো ব্যাখ্যা করতে পারলেই বোঝা যাবে বেসিক ভাষার প্রাথমিক ধাপগুলো পাঠকের রপ্ত হয়েছে।

7

```
10 INPUT L, W          ( এখানে L এর অর্থ দৈর্ঘ্য, আর
20 LET  P=2 * (L+W)      W এর অর্থ প্রস্থ )
30 PRINT  L, W, P
40 END
```

8

```
10 INPUT X
20 LET Y=1+X+X ↑ 2+X ↑ 3
30 PRINT X, Y
40 END
```

9

```
10 INPUT  A, B, C, D, E
20 LET  S=A+B+C+D+E
30 PRINT  A, B, C, D, E
40 PRINT  S
50 END
```

10

```
10 INPUT  A, B
20 LET  C=A+B
30 LET  D=C ↑ 2
40 PRINT  A, B
50 PRINT  D
60 END
```

এখন খুব সহজ কয়েকটা সমস্যা দেওয়া যেতে পারে যা সমাধান করতে পাঠক বেসিক প্রোগ্রাম ব্যবহার করতে পারে।

১। প্রমাণ কর যে $(a+b)(a-b)=a^2+b^2$

২। কোনও বৃত্তের ব্যাসার্ধ r হলে তার ক্ষেত্রফল কত?

৩। প্রমাণ কর যে $(a+b)^3=a^3+3a^2b+3ab^2+b^3$

৪। a বাহু বিশিষ্ট কোনও ঘনকের আয়তন কত?

ওপরের সমস্যাগুলোর মধ্যে তৃতীয়টির বেসিক প্রোগ্রাম লিখে দেওয়া হল :

11

```
10 INPUT  A, B
20 LET  C=A+B
30 LET  D=C ↑ 3
40 LET  X=A ↑ 3+3*(A ↑ 2)*B+3*A*(B ↑ 2)+B ↑ 3
```

```
50 PRINT D
60 PRINT X
70 END
```

ওপরের প্রোগ্রামটিতে A, B-র মান কম্পিউটারকে জানিয়ে দিলে দেখা যাবে যে কম্পিউটারের জানানো উত্তরে D এবং X-এর মান সমান। অতএব এই ভাবেই $(a+b)^3$-এর সূত্রটি প্রমাণিত হল।

বেসিক প্রোগ্রামে ব্যবহৃত পাঁচটি গাণিতিক সংকারক সম্বন্ধে প্রোগ্রামারকে সতর্ক হতে হবে। সাধারণ সরল অঙ্কে যেমন গুণ-ভাগের কাজ আগে করতে হয়, আর যোগ-বিয়োগের কাজ করতে হয় পরে, তেমনি বেসিক প্রোগ্রামের বেলাতেও কোন্ অপারেটর আগে কাজ করবে, আর কোন্‌টিই বা পরে, তার নির্দিষ্ট নিয়ম আছে।

ধরা যাক, বেসিক ভাষায় একটি লাইন এই রকম :

LET A=4*P−7*Q

এর মানে কি $a=4p-7q$, না $a=4(p-7q)$?

আবার যদি থাকে

LET D=A/B*C

তাহলে এর অর্থ $d=(a\div b)\times c$, নাকি $d=a\div(b\times c)$? এই সব সমস্যা থেকে মুক্তি পেতেই বেসিক ভাষার অপারেটরগুলোর গুরুত্বের নির্দিষ্ট শ্রেণীবিভাগ করা হয়েছে। অর্থাৎ নির্দিষ্ট নিয়ম মেনে তবেই অপারেটরগুলো বিভিন্ন গাণিতিক ধাপের কাজ সম্পন্ন করে। কোনও সমীকরণে সবচেয়ে আগে সম্পন্ন হয় ঘাতের (↑) কাজ। সমীকরণের যাবতীয় ঘাতের কাজ সম্পন্ন হয়ে গেলে তবেই শুরু হয় গুণ (*) ও ভাগের (/) কাজ। সমীকরণের বাঁ দিক থেকে এগোলে গুণ বা ভাগ যে-চিহ্নটি আগে পড়বে তার কাজটিই আগে সম্পন্ন হবে। আর যোগ (+) ও বিয়োগের (−) বেলাতেও ঠিক গুণ-ভাগের মতোই নিয়ম। তবে যোগ-বিয়োগের কাজ করা হয় সবার শেষে—ঠিক পাটিগণিতের সরল অঙ্কের মতো।

সুতরাং এবারে আমরা সুনিশ্চিতভাবে বলতে পারি যদি কোনও নির্দেশ থাকে

LET A=4*P−7*Q

তবে তার সঠিক অর্থ হল, $a=4p-7q$

আবার

LET D=A/B*C

এর অর্থ হল $d=(a\div b)\times c$

কোনও বেসিক প্রোগ্রাম লেখার সময়ে অপারেটর ব্যবহারের এই নিয়মগুলো ঠিক ঠিক ভাবে অবশ্যই মেনে চলতে হবে। এছাড়া গুণ চিহ্নকে কখনও উহ্য রাখা চলবে না। যেমন, বীজগণিতের একটি ব্যঞ্জক বা এক্সপ্রেসন যদি হয়

$$3(2a+4b)$$

তাহলে বেসিক ভাষায় রূপান্তরিত করে এটিকে লিখতে হবে

3*(2*A+4*B)

অর্থাৎ, গুণ চিহ্নগুলোকে স্পষ্টভাবে লিখতে হবে—উহ্য রাখলে চলবে না। যদি আমরা ওপরের ব্যঞ্জকটিকে বেসিকে লিখি

3(2*A+4*B) বা 3*(2A+4B)

তাহলে ভুল হবে।

বেসিক ভাষার কয়েকটি মূল শব্দ

এ-পর্যন্ত যে-কটি বেসিক প্রোগ্রামের সঙ্গে আমাদের পরিচয় ঘটেছে তার মধ্যে কয়েকটি বিশেষ শব্দের ব্যবহার লক্ষ্য করার মতো। শব্দগুলো হল, LET, INPUT, PRINT, READ, DATA ও END। এই শব্দগুলোকে আমরা বলব কী-ওয়ার্ড বা মূল শব্দ। বেসিক ভাষায় এরকম অনেক মূল শব্দ আছে। এই মূল শব্দগুলো কোনও বেসিক প্রোগ্রামে গুরুত্বপূর্ণ অর্থ বহন করে।

যে-কটি উদাহরণ আমরা দেখেছি তাতে মূল শব্দগুলোকে বড় হাতের অক্ষরে বা ক্যাপিটাল হরফে প্রকাশ করা হয়েছে। সাধারণত বেসিক প্রোগ্রামে সব অক্ষরই বড় হাতের হরফে লেখা উচিত। তবে আজকাল যে সব পার্সোনাল কম্পিউটার খুব জনপ্রিয় তাতে প্রোগ্রাম লেখার সময়ে ছোট হাতের হরফ বা বড় হাতের হরফ যাই ব্যবহার করা হোক না কেন, কম্পিউটার ঠিক নির্দেশ মেনে তার কাজ করে যাবে। আবার প্রোগ্রামারের লেখা নির্দেশগুলো যখন কম্পিউটারকে ছাপাতে বলা হবে বা ভি. ডি. ইউ.-তে দেখাতে বলা হবে তখন কম্পিউটার কিন্তু সব অক্ষরই বড় হাতের হরফে দেখাবে। অর্থাৎ, কী-বোর্ডে বোতাম টিপে প্রোগ্রামার ছোট-বড় যে কোনও হরফেই প্রোগ্রামের মূল শব্দগুলো লিখুক না কেন কম্পিউটার সেটিকে তার স্মৃতিতে শুধুমাত্র বড় হাতের হরফেই সঞ্চয় করে রাখে।

LET

এবারে LET মূল শব্দটি নিয়ে আলোচনা করা যাক।

বেসিক ভাষায় LET মূল শব্দটি ব্যবহার করে যে-নির্দেশ দেওয়া হয় তার

অর্থ কোনও ভ্যারিয়েব্‌ল্‌ বা চলরাশির ওপর নির্দিষ্ট কোনও মান আরোপ করা। এই নির্দেশের সাহায্যে কোনও চলরাশির সংজ্ঞা নিরূপণ করা যায়।

যেমন,

```
10  LET  X=5·0
20  LET  X1=A
30  LET  C=A+B
40  LET  Y=1+X+X↑2+X↑3
50  LET  P=2*(L+W)
60  LET X2=2+3*11
```

ওপরের ছ'টি উদাহরণে সমান চিহ্নের (=) ঠিক বাঁ দিকে যে-অক্ষর বা অক্ষরগুলো লেখা হয়েছে সেটাই হল চলরাশি। আর সমান চিহ্নের ডান দিকে যে-সংখ্যা বা সমীকরণ বা রাশি লেখা হয়েছে তা হল সেই সেই নির্দেশের চলরাশির ওপরে আরোপিত মান।

যেমন, প্রথম নির্দেশে চলরাশি X, এবং '=' চিহ্নের দ্বারা তার ওপর আরোপিত মান 5·0। অর্থাৎ, এই নির্দেশটি পেলে কম্পিউটার তার স্মৃতি-কোষের একটি নির্দিষ্ট ঠিকানা বা অ্যাড্রেসকে X নামে চিহ্নিত করে সেখানে সঞ্চিত করে রাখবে 5·0 সংখ্যাটি। প্রোগ্রামের পরবর্তী কোনও অংশে যখন X '=' চিহ্নের ডানদিকে ব্যবহার করা হবে তখন কম্পিউটার X-এর মান হিসেবে 5·0 সংখ্যাটিকে বসিয়ে যাবতীয় হিসেব নিকেশের কাজ করবে।

বেসিক ভাষায় কোনও চলরাশির নাম দুটি অক্ষরের বেশি হলে চলবে না। যেমন, ওপরের উদাহরণগুলোতে চলরাশির নাম হিসেবে ব্যবহার করা হয়েছে X, X1, C, Y, P, X2 ইত্যাদি। লক্ষ্য করলে দেখা যাবে, কোনও চলরাশির নাম দু-অক্ষরের হলে দ্বিতীয় অক্ষরটি অবশ্যই কোনও সংখ্যা হতে হবে। তবে বেসিক ভাষার আধুনিক কোনও কোনও সংস্করণে এর ব্যতিক্রম লক্ষ্য করা যায়।

বেসিক ভাষায় চলরাশি দু-রকমের। একধরনের চলরাশি হল সাংখ্যিক চলরাশি বা নিউম্যারিক ভ্যারিয়েব্‌ল্‌—ওপরের উদাহরণে যে-সব চলরাশি দেখানো হয়েছে। এছাড়া অন্য আর এক ধরনের চলরাশি আছে যাকে বলা হয় স্ট্রিং ভ্যারিয়েব্‌ল্‌ বা সারি চলরাশি। এই ধরনের চলরাশির নামকরণ করা হয় দু-অক্ষরে। তার মধ্যে শেষ অক্ষরটি সবসময়েই ডলার চিহ্ন ($)। যেমন, A$, X$, P$ ইত্যাদি।

যদি লেখা হয়,

```
10  LET K$="BASIC"
```

তাহলে এর অর্থ হল, স্মৃতিকোষের K$ চিহ্নিত ঠিকানায় BASIC কথাটির বাইনারি রূপ সঞ্চিত থাকবে। যদি আমরা এর পরেই নির্দেশ দিই

```
20 PRINT K$
```

তাহলে ভি. ডি. ইউ.-তে BASIC লেখাটি ফুটে উঠবে। অর্থাৎ, K$ নামক সারি চলরাশির 'মান' হিসেবে BASIC কথাটিই কম্পিউটার একেবারে নির্দিষ্ট করে রেখেছে।

সারি চলরাশির মান সর্বদা কোটেশান চিহ্নের মধ্যে লিখতে হয়। কোটেশান চিহ্নের মধ্যে যে যে অক্ষর ঠিক যেভাবে লেখা হবে, সারি চলরাশি সর্বদা ঠিক সেইভাবেই তার 'মান'টিকে উপস্থাপিত করবে।

এবারে LET মূল শব্দটি ব্যবহার করে সারি চলরাশি সংক্রান্ত কয়েকটি বেসিক ভাষার নির্দেশ লেখা হল :

```
10 LET X$="I KNOW BASIC"
20 LET B$=C$
30 LET A$="123+456-789"
40 LET N$="N$=12·056"
50 LET P$="WHAT IS YOUR NAME ?"
60 LET Q$="3781"
```

ওপরের উদাহরণগুলোর প্রত্যেকটিতে ডানদিকের কোটেশান চিহ্নের মধ্যে লেখা অক্ষর, সংখ্যা বা বিশেষ চিহ্নগুলোই হল বিভিন্ন সারি চলরাশির 'মান'। এই 'মান'গুলোকে বলা হয় স্ট্রিং বা সারি। কোনও সারিতে যে-কোনও অক্ষর, সংখ্যা, শূন্যস্থান, বা চিহ্ন—যেমন, +, −, *, $, ;, /, =, ইত্যাদি—পর পর সাজানো থাকতে পারে। তবে তার মধ্যে কোনও কোটেশান চিহ্ন থাকা চলবে না। যে-কোনও স্ট্রিং বা সারি কোনও গাণিতিক মান নির্দেশ করে না। যেমন ওপরের উদাহরণে 30 ও 60 নম্বর নির্দেশ। সেখানে সমীকরণ চিহ্নের ডান দিকের সংখ্যাগুলো নেহাতই সংখ্যার 'ছবি' মাত্র, কোনও সাংখ্যিক মান তাদের নেই। ফলে কোনওরকম গাণিতিক হিসেব নিকেশ তাদের সাহায্য নিয়ে করা সম্ভব নয়। সাধারণত কোনও সারিতে 15টি চিহ্ন, অক্ষর বা সংখ্যা ব্যবহার করা যায়।

LET সংক্রান্ত নির্দেশের সঙ্গে যে-সমান চিহ্ন ব্যবহার করা হয় তার সঙ্গে সমীকরণের সমান চিহ্নের তফাৎ রয়েছে। সমীকরণের ক্ষেত্রে '=' চিহ্নের বাঁ দিকের রাশি ডান দিকের রাশির সঙ্গে সমান। কিন্তু LET ব্যবহৃত নির্দেশের

ক্ষেত্রে '=' চিহ্নের বাঁ দিকের চলরাশি তার জন্য নির্দিষ্ট স্মৃতিকোষে সমান চিহ্নের ডান দিকের রাশিটিকে নিজের 'মান' হিসেবে জমা রাখবে। যেমন,

12

```
10  LET A=3·0
20  LET A=A+2·0
30  PRINT A
40  END
```

ওপরের প্রোগ্রামটিতে 20 নম্বর নির্দেশটি লক্ষ্য করার মতো। সাধারণ কোনও সমীকরণের ক্ষেত্রে $a=a+2$ লিখলে তা হবে অর্থহীন। কিন্তু ওপরের বেসিক প্রোগ্রামে 20 নম্বর নির্দেশটি মোটেই অর্থহীন নয়। প্রোগ্রামটা একটু বিশ্লেষণ করলেই ব্যাপারটা বোঝা যাবে।

প্রোগ্রামের প্রথম নির্দেশ—অর্থাৎ, 10 নম্বর নির্দেশ—পাওয়ামাত্রই কম্পিউটার A নামক সাংখ্যিক চলরাশিটির মান হিসেবে 3·0 সংখ্যাটি একটি বিশেষ স্মৃতিকোষে জমা রাখবে। তারপর 20 নম্বর নির্দেশ পেলেই সে সেই একই স্মৃতিকোষে A+2 বা 5·0 জমা করবে। এর ফলে আগে সঞ্চিত মান 3·0 কিন্তু মুছে যাবে কম্পিউটারের 'মগজ' থেকে। সুতরাং 30 নম্বর নির্দেশে পেঁছে কম্পিউটার যখন A চলরাশির মান 'ছাপতে' চাইবে তখন ভি. ডি. ইউ.-তে ফুটে উঠবে 5·0, আগের মান 3·0 নয়। ওপরের প্রোগ্রামটি RUN করালেই ব্যাপারটা হাতেনাতে ধরা পড়বে।

এবারে একটা ছোট প্রোগ্রাম লেখা যাক। এতে মূলত LET ও PRINT ব্যবহার করা হয়েছে।

ধরা যাক কোনও কলেজে ছাত্র-প্রতিনিধি নির্বাচনের জন্য ভোট নেওয়া হল। তিনজন প্রার্থী নির্বাচনে দাঁড়াল—শঙ্কর, অশোক ও প্রশান্ত। শঙ্কর পেল 372টি ভোট, অশোক পেল 188টি আর প্রশান্ত 240টি। আমরা জানতে চাই মোট কতজন ছাত্র ভোট দিয়েছিল এবং কে শতকরা কত ভোট পেয়েছে। এর বেসিক প্রোগ্রামটা নিচে দেওয়া হল ঃ

13

```
10  LET S=372
20  LET A=188
30  LET P=240
40  LET T=S+A+P
50  PRINT "TOTAL NO. OF VOTES="; T
60  PRINT "% FOR SHANKAR="; (S/T)*100; "%"
70  PRINT "% FOR ASHOK="; (A/T)*100; "%"
```

```
80 PRINT "% PRASANTA="; (P/T)*100; "%"
90 END
```

প্রোগ্রামটি RUN করালেই কম্পিউটার উত্তর দেবে ঃ

```
RUN
TOTAL NO. OF VOTES= 800
% FOR SHANKAR= 46·5%
% FOR ASHOK= 23·5%
% FOR PRASANTA= 30%
```

LET মূল শব্দের আলোচনা এখানেই শেষ করে আমরা চলে যাব পরবর্তী মূল শব্দ INPUT-এ।

INPUT

আগের উদাহরণগুলো (প্রোগ্রাম নং 2, 3, 4, 7, 8, 9, 10 এবং 11) দেখলেই INPUT মূল শব্দটির তাৎপর্য এবং ব্যবহারের কায়দা সহজেই বোঝা যাবে। INPUT মূল শব্দের সাহায্যে কম্পিউটারকে চলরাশির মান জানানো হয়। এই চলরাশি সাংখ্যিক অথবা সারি চলরাশি হতে পারে। কোনও প্রোগ্রাম নিয়ে কাজ করার সময় কম্পিউটার INPUT মূল শব্দ ব্যবহারকারী নির্দেশটি থেকেই জানতে পারে, কোন্ কোন্ চলরাশির মান তাকে জানানো হবে। প্রোগ্রাম চালানো হলেই সেই চলরাশিগুলোর মান জানতে চেয়ে কম্পিউটার ভি. ডি. ইউ.-তে জিজ্ঞাসার চিহ্ন (?) ফুটিয়ে তোলে। প্রত্যেকটি চলরাশির মান না জানানো পর্যন্ত কম্পিউটার প্রোগ্রামটি সম্পন্ন করবে না।

নিচে INPUT ব্যবহারকারী কয়েকটি নির্দেশের উদাহরণ দেওয়া হল ঃ

```
10 INPUT A, B
20 INPUT P$, A, B$, X, Y
30 INPUT R$
40 INPUT Z
```

কোনও একটি INPUT নির্দেশের সাহায্যে যদি একাধিক চলরাশির মান জানানো হয় তাহলে চলরাশিগুলোকে 'কমা' চিহ্নের সাহায্যে আলাদা করে বোঝাতে হবে। যেমন, 20 নম্বর নির্দেশে পাঁচটি চলরাশির মান জানানো হয়েছে। এই নির্দেশ ব্যবহার করে কোনও প্রোগ্রাম RUN করালেই কম্পিউটার ভি. ডি. ইউ.-তে একটি ? চিহ্ন ফুটিয়ে তুলবে। তখন কী-বোর্ডের মাধ্যমে প্রোগ্রামারকে চলরাশিগুলোর মান জানাতে হবে এইভাবে ঃ

"INDIA", 10·5, "BASIC", 2·3, 7·0 RET

এখানে লক্ষ্য করা দরকার, প্রোগ্রামের নির্দেশে চলরাশিগুলো পর পর যেভাবে আছে ঠিক সেইভাবেই তাদের মানগুলো কম্পিউটারকে জানানো হয়েছে। যেমন, প্রথমে একটি সারি চলরাশি, তারপর একটি সাংখ্যিক চলরাশি ইত্যাদি। মানগুলো কম্পিউটারকে জানাতে গেলে সেগুলো কী-বোর্ডে টাইপ করার পর রিটার্ন বোতামটি অবশ্যই টিপতে হবে।

সারি চলরাশির মান জানানোর সময়ে কোটেশান চিহ্ন না দিলেও চলে। কিন্তু তার মধ্যে যদি কোনও শূন্যস্থান থাকে বা কমা চিহ্ন থাকে তাহলে কোটেশান চিহ্ন অবশ্যই দিতে হবে। যেমন নিচের উদাহরণটি ঃ

"BASIC, FORTRAN IV"

বেসিক ভাষার DATA নির্দেশের ক্ষেত্রেও এই নিয়মটি প্রযোজ্য।

PRINT, LPRINT

কম্পিউটারের ভি. ডি. ইউ.-তে কোনও কিছু লেখানোর কাজ সম্পন্ন করে PRINT নির্দেশ। একটি PRINT নির্দেশের সাহায্যে একাধিক জিনিস লেখা যেতে পারে, তবে লিখতে চাওয়া প্রত্যেকটি জিনিসের মাঝে কমা অথবা সেমিকোলন চিহ্ন অবশ্যই থাকতে হবে। নিচে কয়েকটি PRINT নির্দেশের উদাহরণ দেওয়া হল ঃ

```
10 PRINT A, B
20 PRINT "NAME :" ; N$
30 PRINT "X=" ; X
40 PRINT A ; B ; C ; D
50 PRINT 5 * 2
60 PRINT X↑2 / 10 ; X
70 PRINT
```

এই উদাহরণগুলোকে এবারে সংক্ষেপে ব্যাখ্যা করা যাক।

10 নম্বর নির্দেশটি আমাদের ইতিমধ্যেই সুপরিচিত। এই নির্দেশ পেলে কম্পিউটার সাংখ্যিক চলরাশি A ও B-এর মান একই লাইনে নির্দিষ্ট দূরত্বে ফুটিয়ে তুলবে। যেমন, যদি A=0·0031 এবং B=−1·2 হয় তাহলে ভি. ডি ইউ.-তে আমরা দেখব,

0·0031 −1·2

20 নম্বর নির্দেশে যদি N$ নামের সারি চলরাশির মান হয় "BASIC", তাহলে ভি. ডি. ইউ.-তে দেখা যাবে,

NAME : BASIC

এখানে সেমিকোলন চিহ্নের অর্থ "NAME ঃ" এবং N$-এর মধ্যে কোনও ফাঁক অথবা শূন্যস্থান থাকবে না। তেমনি একইভাবে 30 নম্বর নির্দেশে X-এর মান যদি 5 হয় তাহলে ভি. ডি. ইউ.-তে ফুটে উঠবে,

X= 5

40 নম্বর নির্দেশে A, B, C ও D-এর মান লেখা হয়ে যাবে পর পর। 50 নম্বর নির্দেশ পেলে কম্পিউটার লিখবে 10 — যা হল কিনা 5 ও 2 এর গুণফল। 60 নম্বর নির্দেশে X-এর মান যদি ৪ হয় তাহলে ভি. ডি. ইউ.-তে দেখা যাবে,

6·4 8

এখানে 6·4 ও 8 এর মধ্যে যে শূন্যস্থান দেখা যাচ্ছে তা হল প্রথম ও দ্বিতীয় চলরাশির জন্য কম্পিউটারের নির্দিষ্ট করা জায়গার অব্যবহৃত শূন্যস্থান। যেমন, সংখ্যা দুটি যদি 6·40123 এবং −8 হত, তাহলে ভি. ডি. ইউ.-তে আমরা হয়ত দেখতে পেতাম,

6·40123−8

অবশ্য কোনও কোনও কম্পিউটারে ভি. ডি. ইউ.-র দুটি সংখ্যার মধ্যে সর্বদাই নির্দিষ্ট ফাঁক রাখার ব্যবস্থা করা আছে।

70 নম্বর নির্দেশের অর্থ এক লাইন শূন্যস্থান। কারণ কম্পিউটারে যে-কোনও নতুন PRINT নির্দেশ পালিত হয় একটি নতুন লাইন থেকে। যেমন,

```
10 PRINT "BASIC IS"
20 PRINT "WONDERFUL !"
```

কম্পিউটার এই নির্দেশ দুটি পেলে আমরা উত্তর হিসেবে পাব

```
BASIC IS
WONDERFUL !
```

যদি আমরা নির্দেশগুলো একটু অন্যভাবে লিখতাম, তাহলে ব্যাপারটা হত অন্যরকম। যেমন,

```
10 PRINT "BASIC IS"
20 PRINT
30 PRINT "WONDERFUL !"
```

এই নির্দেশগুলোর ফলাফল হিসেবে আমরা পাব

BASIC IS

…………… এক লাইন শূন্যস্থান।

WONDERFUL !

যদি ওপরের উদাহরণের 10 নম্বর নির্দেশে শেষ চলরাশি B-এর পরে একটি কমা থাকে, তাহলে 10-এর পরবর্তী কোনও PRINT নির্দেশ থাকলে তার ফলাফলগুলোও B-এর পরে একই লাইনে ফুটে উঠবে। 40 নম্বর নির্দেশে D-এর পরে একটি সেমিকোলন দিলেও ঠিক একই ঘটনা ঘটবে এবং পরের লাইনের ফলাফল ফুটে উঠবে D-এর একেবারে গায়ে। তবে ভি. ডি. ইউ.-তে যতক্ষণ এক লাইনে জায়গা পাওয়া যাবে ততক্ষণই উত্তরগুলো একই লাইনে ফুটে উঠবে। তার পর সেগুলো লেখা হয়ে যাবে পরের লাইনে। বেসিকের কোনও কোনও সংস্করণে দুটি চলরাশির মধ্যে কমা চিহ্ন থাকলে তাদের মান ভিন্ন দুটি লাইনে ফুটে ওঠে। অর্থাৎ উদাহরণের 10 নম্বর নির্দেশ পেলে A ও B-এর মান পর পর দু-লাইনে 'ছাপা' হবে।

বেসিক ভাষার PRINT নির্দেশ সব সময়ে ভি. ডি. ইউ.-তে লেখার কাজ করে। কিন্তু প্রিণ্টারের মাধ্যমে কোনও কিছু কাগজে ছাপাতে চাইলে তার তার জন্য বেসিক নির্দেশ হল LPRINT। কোনও প্রোগ্রামে PRINT-এর বদলে LPRINT ব্যবহার করলে নির্দিষ্ট লেখাগুলো ভি. ডি. ইউ.-তে ফুটে না উঠে ছাপা হয়ে যাবে প্রিণ্টারে কাগজে।

READ—DATA, RESTORE

READ নির্দেশ অনেকটা INPUT নির্দেশের মতোই। INPUT মূল শব্দ ব্যবহার করলে বিভিন্ন চলরাশির মান কী-বোর্ডের মাধ্যমে কম্পিউটারকে জানানো হয় প্রোগ্রাম RUN করানোর পর। কিন্তু READ মূল শব্দ ব্যবহার করলে তার সঙ্গে DATA মূল শব্দ ব্যবহার করতেই হবে। READ ব্যবহারাকরী নির্দেশে থাকবে বিভিন্ন চলরাশির নাম—যেমন, X, A, B$, C, N$ ইত্যাদি, আর DATA ব্যবহারকারী নির্দেশে থাকবে READ নির্দেশে লেখা বিভিন্ন চলরাশির মান—যেমন, 5·O, –7, BASIC, 2·31, NAME ইত্যাদি। অর্থাৎ, READ ও DATA একটি বেসিক প্রোগ্রামেরই অঙ্গ। এই দুটি মূল শব্দের সাহায্য নিয়ে কোনও বেসিক প্রোগ্রাম লিখলে সেই প্রোগ্রাম RUN করানোর পর কম্পিউটারকে আর কোনও চলরাশির মান কী-বোর্ডের মাধ্যমে জানাতে হয় না। উদাহরণ হিসেবে 3 ও 5 নম্বর প্রোগ্রাম লক্ষ্যণীয়। যে-সব বেসিক প্রোগ্রামে প্রচুর চলরাশির মান কাজে লাগানো হয় সেই সব প্রোগ্রামের ক্ষেত্রে

READ—DATA নির্দেশ INPUT নির্দেশের চেয়ে অনেক বেশি সুবিধেজনক। নিচে কয়েকটি READ—DATA নির্দেশের উদাহরণ দেওয়া হল ঃ

```
10  READ X, Y, Z, A$
20  DATA 5, -3, 11, ASHIM

10  READ A, B
20  READ C, P$, Q$
30  DATA -2, 7·3, 1·45, PRANAB
40  DATA SUSHANTA

10  READ A, B, C, D
20  DATA 2, 4
30  READ E, F, G
40  DATA 5, 7, 9
50  DATA -13, 3
```

ওপরের তিনটি উদাহরণ থেকে বোঝা যাচ্ছে READ নির্দেশে প্রতিটি চলরাশির নাম এবং DATA নির্দেশে প্রতিটি চলরাশির মান-এর মধ্যে কমা চিহ্ন দিতে হবে। READ নির্দেশে চলরাশির নাম যেভাবে পর পর থাকবে, DATA নির্দেশে তাদের মানগুলো কম্পিউটার ঠিক সেই ক্রমানুসারে গ্রহণ করবে। যেমন প্রথম উদাহরণের ক্ষেত্রে

X = 5, Y = −3, Z = 11 এবং A$ = ASHIM

দ্বিতীয় উদাহরণে দুটি READ ও দুটি DATA নির্দেশ ব্যবহার করা হয়েছে। তা না করে লাইন চারটিকে এইভাবেও লেখা যেত ঃ

```
10  READ A, B, C, P$, Q$
20  DATA -2,7·3, 1·45, PRANAB, SUSHANTA
```

যদি একটি নির্দেশ খুব লম্বা হয়ে যায় তাহলে সেটিকে একাধিক নির্দেশে ভেঙে নেওয়া হয়—যেমনটি দ্বিতীয় বা তৃতীয় উদাহরণে করা হয়েছে। এখানে আরও একটা জিনিস লক্ষ্যণীয় ঃ READ বা DATA নির্দেশের সংখ্যা এক বা একাধিক যাই হোক না কেন, কম্পিউটার চলরাশির নাম যেমন যেমন ভাবে পাবে ঠিক সেই ক্রমানুসারে তাদের মান DATA নির্দেশ থেকে একের পর এক গ্রহণ করে স্মৃতিকোষে সঞ্চয় করে রেখে দেবে। চলরাশির মান-এর সংখ্যার তুলনায় যদি তাদের নাম বেশি থাকে তাহলে কম্পিউটার কাজ না করে জানিয়ে

দেবে ভুল হয়েছে। আর যদি ঠিক উল্টো ব্যাপার হয়—অর্থাৎ, চলরাশির নামের তুলনায় তাদের মান-এর সংখ্যা বেশি হয়ে যায়—তাহলে কম্পিউটার বাড়তি মানগুলোকে স্রেফ উপেক্ষা করবে।

তৃতীয় উদাহরণে দুটি READ ও তিনটি DATA নির্দেশ ব্যবহার করা হয়েছে। কিন্তু লক্ষ্য করলে দেখা যাবে এই উদাহরণে মোট চলরাশির নামের সংখ্যা সাত, এবং DATA নির্দেশগুলোতে দেওয়া মোট মান-এর সংখ্যাও ঠিক তাই।

চলরাশি সাংখ্যিক বা সারি, যেরকম হবে, তাদের মানও সেই অনুযায়ী ঠিক করতে হবে। লক্ষ্য রাখতে হবে, READ ও DATA নির্দেশের প্রত্যেকটি চলরাশির নাম ও তাদের নিজের নিজের মান বেসিক ভাষার নিয়ম অনুযায়ী যেন নির্ভুল হয়। সাংখ্যিক চলরাশির নামের সঙ্গে (যেমন প্রথম উদাহরণের X) যেন সারি চলরাশির মান (যেমন প্রথম উদাহরণের ASHIM) না চলে আসে। তাহলে কম্পিউটার প্রোগ্রাম অনুযায়ী কাজ সম্পন্ন না করে জানাবে ঃ ভুল হয়েছে। এছাড়া মনে রাখা দরকার, DATA নির্দেশে কখনও কোনও চলরাশির নাম বা কোনও সূত্র লেখা যায় না।

সাধারণত READ নির্দেশ লেখা হয় কোনও প্রোগ্রামের শুরুতে, আর DATA নির্দেশ লেখা হয় প্রোগ্রামের প্রায় শেষে—END নির্দেশের ঠিক আগে। DATA নির্দেশ অবশ্য প্রোগ্রামের যে-কোনও জায়গাতেই লেখা যেতে পারে। তবে প্রোগ্রামিং-এর সুবিধের জন্য DATA নির্দেশ সবসময় শেষেই লেখা উচিত।

READ ও DATA নির্দেশের সঙ্গে আর একটি মূল শব্দ আলোচনা করা প্রয়োজন। সেটি হল RESTORE। RESTORE নির্দেশ ব্যবহার করে আমরা READ নির্দেশকে পরিচালিত করতে পারি। অর্থাৎ, একটি প্রোগ্রামের READ নির্দেশ সেই প্রোগ্রামের কোন্ DATA নির্দেশ থেকে চলরাশির মান গ্রহণ করবে তা RESTORE নির্দেশ ব্যবহার করে নির্দিষ্ট করে দেওয়া যেতে পারে। যেমন,

14

```
10 READ X, Y
20 RESTORE
30 READ A, B
40 PRINT X; Y; A; B
50 DATA 8, 13
60 END
```

এই প্রোগ্রামটি চালালে কম্পিউটার X এবং Y-এর মান হিসেবে যথাক্রমে

8 ও 13 গ্রহণ করবে। আবার 20 নম্বর RESTORE নির্দেশ অনুযায়ী A এবং B-এর মানও সে যথাক্রমে 8 ও 13 হিসেবে গ্রহণ করবে। ফলে 40 নম্বরের PRINT নির্দেশ অনুযায়ী ভি. ডি. ইউ.-তে ফুটে উঠবে ঃ

8 13 8 13

সুতরাং RESTORE নির্দেশ ব্যবহার করে 50 নম্বর নির্দেশের মান 8 ও 13 আমরা বিভিন্ন চলরাশির মান হিসেবে বার বার ব্যবহার করতে পারি।

কোন্ DATA নির্দেশ থেকে কম্পিউটারকে মান 'পড়তে' হবে তা RESTORE নির্দেশের সাহায্যে নির্দিষ্ট করে দেওয়া যায়। যেমন,

15

```
10 RESTORE 60
20 READ X, Y
30 PRINT X ; Y
40 DATA −1·3, 9
50 DATA 10, 20
60 DATA −5, −25
70 DATA 0·01, 100
80 END
```

এই প্রোগ্রামের উত্তর হবে ঃ

```
RUN
−5 −25
```

কারণ 10 নম্বর লাইনের RESTORE নির্দেশ অনুযায়ী 60 নম্বর লাইনের DATA নির্দেশের মানই কম্পিউটার X ও Y-এর মান হিসেবে গ্রহণ করবে।

15 নম্বর প্রোগ্রামটিকে একাধিক RESTORE নির্দেশ ব্যবহার করে লেখা যেতে পারে।

16

```
10 RESTORE 130
20 READ X, Y
30 PRINT X ; Y ;
40 RESTORE 140
50 READ X, Y
60 PRINT X ; Y ;
70 RESTORE 150
80 READ X, Y
90 PRINT X ; Y ;
```

```
100 RESTORE 160
110 READ X, Y
120 PRINT X ; Y
130 DATA −1·3, 9
140 DATA 10, 20
150 DATA −5, −25
160 DATA 0·01, 100
170 END
```

এই প্রোগ্রামটি চালালে উত্তর পাওয়া যাবে ঃ

```
RUN
−1·3 9 10 20 −5 −25
 0·01 100
```

নিচে RESTORE ব্যবহার করে আর একটি উদাহরণ দেওয়া হল। আশা করি, পাঠক নিজেই উদাহরণটি ব্যাখ্যা করতে পারবে।

17

```
10 READ A, B, C
20 PRINT "TOTAL COST IS :" ;
30 PRINT A+B+C
40 PRINT "SEPARATE COSTS ARE" ;
50 RESTORE
60 PRINT A ; "," ; B ; "," ; C
70 DATA 6, 8, 4
80 END
```

এই প্রোগ্রাম RUN করালে উত্তর পাওয়া যাবে ঃ

```
RUN
TOTAL COST IS : 18
SEPARATE COSTS ARE :   6, 8, 4
```

END

END নির্দেশ কোনও বেসিক প্রোগ্রামের অবিচ্ছেদ্য অঙ্গ। কোনও বেসিক প্রোগ্রাম যে শেষ হয়েছে তা কম্পিউটার বুঝতে পারে END নির্দেশটি দেখে। END নির্দেশ সবসময়েই থাকবে কোনও প্রোগ্রামের একেবারে শেষে এবং তার স্টেটমেণ্ট নাম্বার বা নির্দেশ-সংখ্যা হবে সবচেয়ে বেশি। পাঠককে এ-প্রসঙ্গে আগের উদাহরণগুলো দেখতে অনুরোধ করছি। তাহলে END-এর ব্যবহার

স্পষ্ট হয়ে যাবে। পর পর অনেকগুলো বেসিক প্রোগ্রাম কম্পিউটারে লিখলে END নির্দেশটির সাহায্যেই কম্পিউটার বুঝতে পারবে কখন একটি প্রোগ্রাম শেষ হয়ে নতুন আর একটি প্রোগ্রাম শুরু হচ্ছে।

ফ্লো-চার্ট বা প্রবাহ-চিত্র

কম্পিউটারের নিজস্ব কোনও 'বুদ্ধি' নেই। প্রোগ্রামের মাধ্যমে তাকে যেমন যেমন নির্দেশ দেওয়া হবে সে ঠিক সেইভাবেই কাজ করবে। নির্দেশের মধ্যে যদি ব্যাকরণ অথবা যুক্তির ভুল থাকে তাহলে কম্পিউটার প্রোগ্রামারকে এরর মেসেজ বা ভ্রম-সংবাদ জানিয়ে চুপচাপ নিষ্কর্মা হয়ে বসে থাকবে। প্রোগ্রামের ব্যাকরণ নিয়ে আমরা সামান্য আলোচনা করেছি, কিন্তু প্রোগ্রামের যুক্তিগত দিকটার কথা উল্লেখ করা হয়নি। এখন আমরা সেই দিকটাই একটু খুঁটিয়ে দেখার চেষ্টা করব।

উদাহরণ হিসেবে 1 নম্বর প্রোগ্রামটির কথাই ধরা যাক।

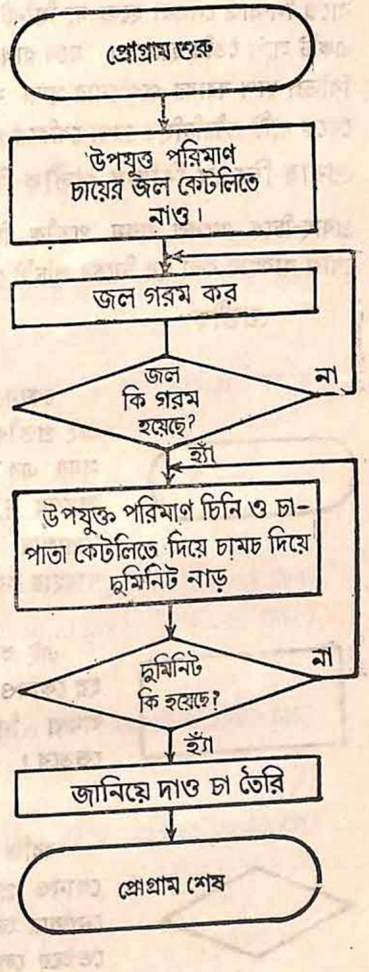
এই ছোট্ট প্রোগ্রামটির প্রতিটি নির্দেশ পালন করে কম্পিউটার কী ভাবে যুক্তি মেনে ধাপে ধাপে কাজটি সম্পন্ন করেছে, তার বিশদ বিবরণ আমরা দেখতে পাচ্ছি প্রোগ্রামটির ঠিক ওপরের ছবিটা থেকে। এই ছবিকেই বলা হয় ফ্লো-চার্ট বা প্রবাহ-চিত্র (পৃষ্ঠা 52)। ত্রিভুজের ক্ষেত্রফল বের করার কোনও অঙ্ক কষতে গেলে আমরা যেভাবে ধাপে ধাপে এগোতাম এখানে ঠিক একইভাবে প্রয়োজনীয় নির্দেশগুলো পর পর দেওয়া হয়েছে কম্পিউটারকে। এই ধাপগুলো ওলট-পালট করে দিলে আমরা যেমন অঙ্কটা কষতে পারতাম না, তেমনি কম্পিউটারও ব্যর্থ হত উত্তর বের করতে। যেমন, 30 নম্বরের নির্দেশটা যদি আমরা 10 নম্বর নির্দেশের ঠিক পরেই দিতাম এবং বেসিক ভাষার ব্যাকরণ অনুযায়ী নির্দেশটার নম্বর 15 করে দিতাম, তাহলে প্রোগ্রামটা দাঁড়াত এই রকম :

```
10    INPUT B, H
15    PRINT A
20    LET A=(1/2)*B*H
40    END
```

সন্দেহ নেই যে পৃথিবীর কোনও কম্পিউটারই এই প্রোগ্রাম পালন করতে পারবে না। কারণ এখানে B ও H-এর মান পড়ার পরই ক্ষেত্রফল A-র মান 'ছাপতে' বলা হচ্ছে। অথচ B ও H-এর সাহায্যে সূত্র ($A=\frac{1}{2}\times B\times H$) ব্যবহার করে A-র মান নির্ণয় করা হচ্ছে তার পরে। বুদ্ধি দিয়ে বিচার করে অতি সহজেই আমরা বলতে পারছি, এই প্রোগ্রামটির যুক্তিগত দিকটায় গলদ রয়েছে। প্রোগ্রামটির অবাস্তবতা আরও স্পষ্ট হবে যদি আমরা ওপরের প্রোগ্রাম

অনুযায়ী তার সঠিক প্রবাহ-চিত্রের শেষ দুটি আয়তাকার বাক্সের জায়গা অদল-বদল করে দিই।

যে-প্রোগ্রামটি নিয়ে আমরা আলোচনা করলাম সেটি এতই সহজ এবং তার মাপ এতই ছোট যে প্রোগ্রামের যুক্তিগত দিক খুঁটিয়ে দেখার ব্যাপারটা বাড়াবাড়ি বলে মনে হতে পারে। কিন্তু বড় এবং জটিল প্রোগ্রামের ক্ষেত্রে দেখা যাবে, সব সময়ে প্রথমে বিচার করা উচিত প্রোগ্রামের যুক্তিগত দিক, আর তার পরে খোদ প্রোগ্রাম এবং তার প্রয়োজনীয় ব্যাকরণ। কোনও কম্পিউটার প্রোগ্রামের যুক্তিগত দিক বিচার করার হাতিয়ার হল প্রবাহ-চিত্র।



কম্পিউটারের সাহায্যে কোনও সমস্যার সমাধান করতে হলে সবচেয়ে আগে ভেবে নিতে হবে ঠিক কী কী ধাপ অনুসরণ করে সমস্যাটি সমাধান করা যায়। সেই ধাপগুলোকে বিভিন্ন প্রতীক চিহ্ন ও তীরচিহ্ন ব্যবহার করে যে-ছবির সাহায্যে বোঝানো হয় তাকেই বলা হয় প্রবাহ-চিত্র।

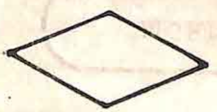
1 নম্বর প্রোগ্রামের প্রবাহ-চিত্রের মতো আর একটা প্রবাহ-চিত্রের উদাহরণ দেওয়া যাক। উদাহরণের সমস্যাটি হল চা তৈরি করা। পাশের প্রবাহ-চিত্রের মাধ্যমে চা তৈরির যুক্তিসঙ্গত প্রতিটি ধাপ দেখান হল।

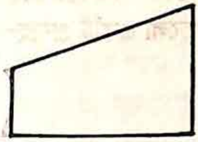
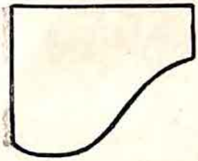
এই প্রবাহ-চিত্রটি এতই সহজ-সরল যে নতুন করে ব্যাখ্যা করার আর কিছু নেই। তবে প্রথম প্রবাহ-চিত্রের সঙ্গে তুলনা করলে দেখা যাবে এই ছবিতে বরফি আকারের এক ধরনের বাক্স রয়েছে যা আগের ছবিটিতে ছিল না। কোনও সমস্যা সমাধানের সময় যদি কোনও একটি ধাপে 'হ্যাঁ' কিংবা 'না' এরকম সিদ্ধান্ত নেবার ব্যাপার থাকে—অর্থাৎ, কোনও প্রশ্ন থাকে—তাহলে এইরকম বরফি আকারের বাক্স ব্যবহার করা হয়। ইংরেজিতে একে বলা হয় 'ডিসিশান বক্স'।

ডিসিশান বক্স বা প্রশ্ন-বাক্সের সিদ্ধান্ত বা উত্তর অনুযায়ী প্রোগ্রামের একটি ধাপ পরবর্তী ধাপ বেছে নেবে। যেমন, 'জল কি গরম হয়েছে?' প্রশ্নটির উত্তর যতক্ষণ 'না' হবে ততক্ষণ পর্যন্ত প্রোগ্রামের পরবর্তী ধাপ হবে 'জল গরম কর'। আর প্রশ্নের উত্তর যদি 'হ্যাঁ' হয়, একমাত্র তখনই গরম জলে চিনি ও চা-পাতা মেশানো যাবে। প্রশ্ন-বাক্সের উত্তর 'না' হলে যে-তীরচিহ্নিত রেখাটি ওপরের 'জল গরম কর' ধাপটিতে ফিরে গেছে এই ধরনের রেখাকে বলা হয় 'লুপ'। প্রবাহ-চিত্রটিতে দ্বিতীয় আর একটি প্রশ্ন-বাক্স রয়েছে। এই প্রশ্ন-বাক্সে সিদ্ধান্ত নেওয়া হচ্ছে দু-মিনিট হয়েছে কি হয়নি, এবং এখানেও আর একটি লুপ তৈরি হয়েছে। মনে রাখতে হবে, প্রবাহ-চিত্র অনুযায়ী প্রোগ্রামের বিভিন্ন ধাপ বরাবর এগোনোর সময় যখনই কোনও প্রশ্ন-বাক্স পড়বে তখনই তা থেকে দুটি তীরচিহ্নিত রেখা বেরিয়ে প্রবাহ-চিত্রে শাখা-পথের সৃষ্টি করবে।

প্রবাহ-চিত্রের বিভিন্ন প্রতীক চিহ্ন ও নিয়ম

প্রবাহ-চিত্রে অনেক রকম প্রতীক চিহ্ন ব্যবহার করা হয়। তার মধ্যে সবচেয়ে বেশি ব্যবহার করা হয় নিচের পাঁচটি প্রতীক ঃ

প্রতীক	অর্থ
	কোনও প্রবাহ-চিত্রের শুরু অথবা শেষ বোঝাতে এই প্রতীক ব্যবহার করা হয়। প্রবাহ-চিত্র গঠনের সময় এর মধ্যে লিখতে হয় 'শুরু' অথবা 'শেষ'। আগের দুটি উদাহরণে বোঝানোর সুবিধের জন্য 'প্রোগ্রাম শুরু' ও 'প্রোগ্রাম শেষ' কথাগুলো ব্যবহার করা হয়েছে।
	এই আয়তাকার বাক্সের প্রতীকটি ব্যবহার করা হয় কোনও কাজ বা নির্দেশ বোঝাতে। সেই কাজ অথবা নির্দেশটি লিখতে হয় প্রতীক চিহ্নটির ভেতরে।
	বরফি চেহারার এই প্রতীকটি ব্যবহার করা হয় কোনও প্রশ্ন করার জন্য অথবা কোনও সিদ্ধান্ত নেওয়ার জন্য। মনে রাখতে হবে, এই প্রতীকের ভেতরে লেখা কোনও প্রশ্নের উত্তর সব সময়েই হবে 'হ্যাঁ' অথবা 'না'—সিদ্ধান্তের ক্ষেত্রে যার অর্থ 'ঠিক' অথবা 'ভুল'।

প্রতীক	অর্থ
	কম্পিউটারের কী-বোর্ডের মাধ্যমে দেওয়া তথ্য বা নির্দেশ।
	কম্পিউটারের ছাপানো তথ্য বা লেখা।

প্রবাহ-চিত্র আঁকার কয়েকটি প্রধান নিয়ম হল :

১। তীর চিহ্নিত প্রবাহ-রেখাগুলো সব সময় ওপর থেকে নিচে নামবে, আর সাধারণত পৃষ্ঠার বাঁ দিক থেকে ডান দিকে এগোবে।

২। যে-প্রবাহ-রেখা ওপর দিকে যাবে তাকে একটি লুপ সম্পূর্ণ করতে হবে।

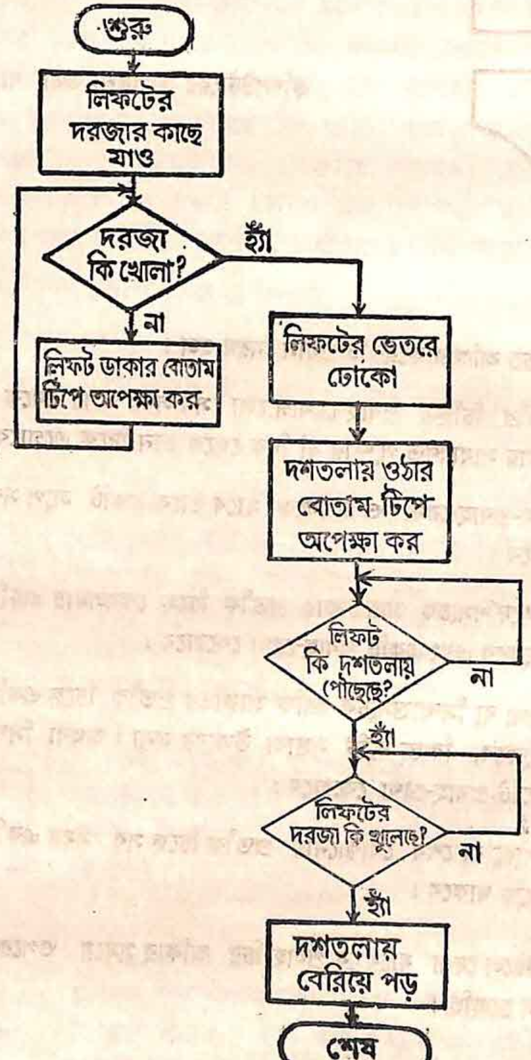
৩। নির্দেশসূচক আয়তাকার প্রতীক চিহ্নে কেবলমাত্র একটি প্রবাহ-রেখা ঢুকবে এবং একটি প্রবাহ-রেখা বেরোবে।

৪। প্রশ্ন বা সিদ্ধান্তসূচক বরফি আকৃতির প্রতীক চিহ্নে একটি প্রবাহ-রেখা ঢুকবে, কিন্তু দুটি সম্ভাব্য উত্তরের জন্য (অথবা সিদ্ধান্তের জন্য) দুটি প্রবাহ-রেখা বেরোবে।

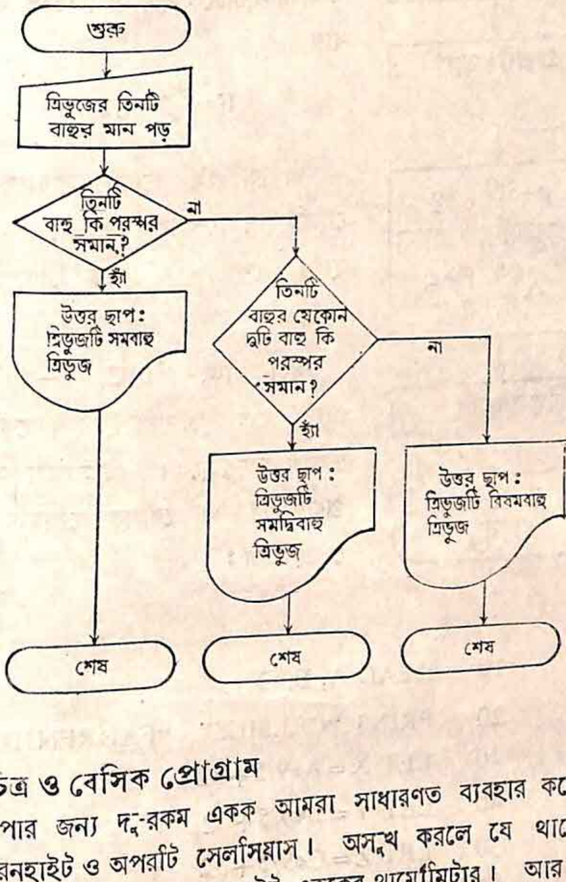
৫। শুরু বা শেষ বোঝানোর প্রতীক চিহ্নে সব সময় একটি প্রবাহ-রেখা যুক্ত থাকবে।

লক্ষ্য করলে দেখা যাবে যে প্রবাহ-চিত্র আঁকার সময়ে ওপরের নিয়মগুলো আমরা মেনে চলেছি।

এবারে নতুন আর একটা উদাহরণ নেওয়া যাক। ধরা যাক, কোনও বাড়িতে গিয়ে কাউকে লিফটে চড়ে সেই বাড়ির দশতলায় পেঁাছতে হবে। লিফটে চড়ে দশতলায় যেতে হলে তাকে ঠিক কী কী কাজ করতে হবে সেগুলো একটি প্রবাহ-চিত্রের মাধ্যমে দেখানো হল ঃ



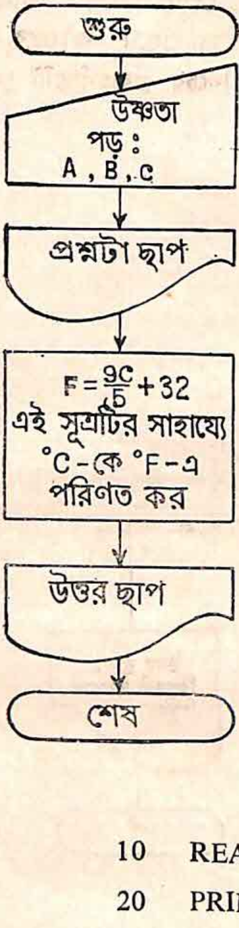
এখন যে উদাহরণটি আমরা আলোচনা করব সেটি জ্যামিতি-সংক্রান্ত একটি সমস্যা। ধরা যাক, কোনও ত্রিভুজের তিনটি বাহু দেওয়া আছে। আমাদের বের করতে হবে, ত্রিভুজটি সমবাহু, সমদ্বিবাহু, না বিষমবাহু। কীভাবে ধাপে ধাপে এই সমস্যার সমাধান করা যায় তা বোঝাতে নিচের প্রবাহ-চিত্রটি দেওয়া হল ঃ



প্রবাহ-চিত্র ও বেসিক প্রোগ্রাম

উষ্ণতা মাপার জন্য দু-রকম একক আমরা সাধারণত ব্যবহার করে থাকি। একটি ফারেনহাইট ও অপরটি সেলসিয়াস। অসুখ করলে যে থার্মোমিটারে আমরা জ্বর মাপি তা হল ফারেনহাইট এককের থার্মোমিটার। আর দৈনন্দিন তাপমাত্রা বোঝাতে সাধারণত সেলসিয়াস এককটি ব্যবহার করা হয়। ফারেনহাইট স্কেলে কোনও উষ্ণতার মান যদি F হয় তাহলে সেলসিয়াস স্কেলে সেই উষ্ণতার মান হবে

$$\frac{5(F-32)}{9}$$



অর্থাৎ, আমরা লিখতে পারি

$$C=\frac{5(F-32)}{9}$$

যেখানে C হচ্ছে সেলসিয়াস স্কেলে উষ্ণতার মান।

ওপরের সূত্রটিকে সামান্য রদবদল করে লেখা যায়

$$F=\frac{9C}{5}+32$$

অর্থাৎ, এই সূত্রটির সাহায্যে আমরা সেলসিয়াস স্কেলের তাপমাত্রাকে ফারেনহাইট স্কেলের এককে প্রকাশ করতে পারি।

ধরা যাক, 50°C, −40°C এবং 80°C কে ফারেনহাইট এককে (°F) প্রকাশ করতে হবে। এই সমস্যা সমাধানের প্রবাহ-চিত্র ও বেসিক প্রোগ্রাম পরপর দেওয়া হল :

18

```
10  READ A, B, C
20  PRINT "CELSIUS", "FAHRENHEIT"
30  LET X=A*9/5+32
40  LET Y=B*9/5+32
50  LET Z=C*9/5+32
60  PRINT A, X
70  PRINT B, Y
80  PRINT C, Z
90  DATA 50, -40, 80
100 END
```

এই প্রোগ্রামটি চালালে উত্তর পাওয়া যাবে ঃ

```
RUN
CELSIUS              FAHRENHEIT
 50                   122
-40                   -40
 80                   176
```

কয়েকটি বেসিক প্রোগ্রাম

বেসিক প্রোগ্রামিং-এর প্রথম পাঠ আমরা শেষ করব দুটি সমস্যা সমাধানের বেসিক প্রোগ্রাম দিয়ে। এই দুটো প্রোগ্রামের জন্য কোনও প্রবাহ-চিত্র এঁকে দেওয়া হল না—সে-দায়িত্ব রইল পাঠকের ওপরে।

১। আকাশবাণী কলকাতা কেন্দ্রের তিনটি বিভাগ আছে ঃ A, B ও C বিভাগ। সকাল সাতটা থেকে রাত দশটা পর্যন্ত প্রত্যেক ঘণ্টায় শতকরা কত জন শ্রোতা C বিভাগের অনুষ্ঠান শোনে ?

এই প্রশ্নের উত্তর বের করতে হলে প্রথমেই জানতে হবে সকাল সাতটা থেকে রাত দশটা পর্যন্ত প্রতি ঘণ্টায় মোট কতজন শ্রোতা রেডিওর অনুষ্ঠান শোনে এবং তার মধ্যে কতজন C বিভাগের অনুষ্ঠান শোনে। তারপর একটি বেসিক প্রোগ্রামের সাহায্যে শতকরা হিসেবের উত্তরটি বের করে ফেলা যাবে।

মনে করি, X = কোনও নির্দিষ্ট ঘণ্টায় মোট শ্রোতার সংখ্যা

আর Y = ওই সময়ে যে-কজন শ্রোতা C বিভাগের অনুষ্ঠান শোনে। তাহলে প্রোগ্রামটি হবে এই রকম ঃ

19

```
10 INPUT X
20 INPUT Y
30 LET Z=(Y/X)*100
40 PRINT "X="; X, "Y="; Y, "Z="; Z; "%"
50 END

RUN
? 218680
? 54670
X=218680   Y=54670   Z=25%
```

প্রোগ্রামটি বার বার চালাতে চাইলে শুধু R U N বোতামগুলো টিপে X ও Y-এর নতুন মান কম্পিউটারকে জানাতে হবে। তাহলেই নতুন উত্তর পাওয়া যাবে Z-এর মান হিসেবে। অর্থাৎ

RUN
? 233660
? 46732
X=233660 Y=46732 Z=20%

এইভাবে কম্পিউটারে একটি প্রোগ্রামকে যতবার খুশি চালানো যেতে পারে।

২। সারা জীবনে একজন মানুষ কত ঘণ্টা ঘুমিয়ে কাটায় ?

এই প্রশ্নের উত্তর বের করতে হলে প্রথমে জানতে হবে মানুষটির বয়েস। সাধারণত কোনও মানুষ দিনের মধ্যে 8 ঘণ্টা ঘুমোয়। অর্থাৎ, তার বয়েসের একের তিন ভাগ সে ঘুমিয়ে কাটায়। সুতরাং, মনে করা যাক, X=মানুষটির বয়েস যত বছর। তাহলে এই সমস্যা সমাধানের বেসিক প্রোগ্রামটি হবে এই রকম ঃ

20

```
Ok
LIST
10   PRINT  " HOW  OLD  ARE  YOU";
20   INPUT  X
30   LET  H = X*365*24
40   PRINT  " DO  YOU  KNOW, YOU  HAVE"
50   PRINT  " SLEPT  FOR";H/3;"HOURS !"
60   END
Ok

RUN
 HOW  OLD  ARE  YOU? 15
 DO  YOU  KNOW, YOU  HAVE
 SLEPT  FOR 43800 HOURS !
Ok
```

ইচ্ছে করলে বিভিন্ন মানুষের বিভিন্ন বয়েসের জন্য এই প্রোগ্রামটি বার বার চালিয়ে প্রয়োজনীয় উত্তর বের করা যেতে পারে। লক্ষ্য করার বিষয়, 10 নম্বর নির্দেশের শেষে সেমিকোলন ব্যবহারের জন্য কম্পিউটার X-এর মান জানতে চেয়ে যে-জিজ্ঞাসা-চিহ্নটি ফুটিয়ে তুলবে সেটা 'ছাপা' হবে YOU শব্দটির ঠিক পরেই—একটি সম্পূর্ণ প্রশ্নবোধক বাক্যের মতো। প্রোগ্রামটি RUN করালে কম্পিউটার যেভাবে উত্তর দেবে তা প্রোগ্রামের নিচেই দেখানো হল। এই প্রোগ্রামটি চালানো হয়েছে IDM-PC কম্পিউটারে।

বেসিক ভাষার প্রাথমিক পরিচয়কে পুঁজি করে এবারে আমরা এগিয়ে যাব পরবর্তী অধ্যায়ে। সেখানে আমরা আলোচনা করব বেসিকের আরও কয়েকটি জরুরী মূল শব্দ নিয়ে।

৫ □ বেসিক প্রোগ্রামিং ঃ দ্বিতীয় পাঠ

বেসিক ভাষার আরও কয়েকটি মূল শব্দ

এখন আমরা বেসিকের আরও কয়েকটি মূল শব্দের অর্থ ও প্রয়োগ নিয়ে আলোচনা করব। আর একই সঙ্গে বিভিন্ন প্রোগ্রাম লিখে তার মাধ্যমে মূল শব্দগুলোর বৈশিষ্ট ব্যাখ্যা করব।

REM

REM মূল শব্দটি প্রকৃতপক্ষে হল ইংরিজি REMARK শব্দের সংক্ষিপ্ত রূপ। ইংরিজি রিমার্ক শব্দের অর্থ হল মন্তব্য। বেসিক প্রোগ্রামে REM-এর ব্যবহারও ঠিক সেই রকমের—অর্থাৎ, বিভিন্ন মন্তব্য লেখার জন্য। যেমন, আগের অধ্যায়ের শেষে যে 20 নম্বর প্রোগ্রামটি আমরা লিখেছি তার শুরুতে খুব সহজেই একটি উপযুক্ত REM নির্দেশ আমরা জুড়ে দিতে পারি ঃ

```
5 REM .. PROGRAM TO CALCULATE HOURS SLEPT
```

এর ফলে প্রোগ্রামটি কী বিষয়ের তা REM নির্দেশটি পড়লেই বোঝা যাবে। যদি কোনও কারণে একটি লাইনে বক্তব্য শেষ না হয় সেক্ষেত্রে একাধিক REM নির্দেশ পরপর ব্যবহার করা যেতে পারে। যেমন

```
5 REM .. PROGRAM TO CALCULATE HOURS SLEPT
6 REM .. BY A MAN IN HIS WHOLE LIFE
```

একটা কথা মনে রাখতে হবে, REM নির্দেশ পেলে কম্পিউটার হিসেব কষার কোনও কাজ করে না—REM নির্দেশের মধ্যে যে কথাগুলো লেখা হয়েছে সেগুলোকে সে প্রোগ্রামের অন্যান্য নির্দেশের মতো শুধু নিজের স্মৃতিকোষে সঞ্চয় করে রাখে। কারণ REM নির্দেশে যা লেখা হয় তা প্রোগ্রামেরই অবিচ্ছেদ্য অঙ্গ। ফলে কম্পিউটার যখন প্রোগ্রামটি প্রিণ্টারের মাধ্যমে কাগজে ছাপিয়ে দেয় তখন REM নির্দেশগুলোও একই সঙ্গে হুবহু ছাপা হয়।

ওপরের উদাহরণে REM নির্দেশে মন্তব্যের অংশটুকু তলায় দাগ টেনে বোঝানো হল। অর্থাৎ, দেখা যাচ্ছে, REM মূল শব্দটি ছাড়া বাকি সবটুকুই প্রোগ্রামারের মন্তব্য।

একটি প্রোগ্রামের যে-কোনও জায়গায় যত খুশি REM নির্দেশ লেখা যায়। কোনও প্রোগ্রাম ঠিক কী কাজ করছে তা চটপট বোঝার জন্য REM নির্দেশ সুবিধেজনক। অনেক সময় খুব বড় বড় বেসিক প্রোগ্রামে প্রোগ্রামের বিভিন্ন অংশের কাজ কী তা প্রোগ্রামের মধ্যেই উল্লেখ করা থাকলে সুবিধে হয়। তখন প্রোগ্রামার প্রয়োজন মতো REM নির্দেশ ব্যবহার করে। একটি প্রোগ্রামের বিভিন্ন উপযুক্ত জায়গায় যদি প্রয়োজনীয় REM নির্দেশ লেখা থাকে তাহলে একজনের করা প্রোগ্রাম পড়ে অন্য কেউ সহজেই তার অর্থ বুঝতে পারে। তাছাড়া প্রোগ্রাম লেখার সময়ে REM নির্দেশ ব্যবহার করার অভ্যেস থাকা ভাল। REM নির্দেশটিকে যাতে স্পষ্টভাবে পড়তে পারা যায় তার জন্য নির্দেশটির আগে-পরে কয়েক লাইন করে শূন্যস্থান রাখতে পারলে ভাল হয়।

GOTO

কোনও বেসিক প্রোগ্রামের প্রতিটি নির্দেশ কম্পিউটার পালন করে ধাপে ধাপে। অর্থাৎ, প্রথম নির্দেশ থেকে শুরু করে একে একে সব নির্দেশ পেরিয়ে সে পেঁছে যায় শেষ নির্দেশে। কিন্তু অনেক সময় কোনও প্রোগ্রামের একটি নির্দেশ থেকে অন্য কোনও একটি নির্দেশে 'লাফ' দেবার প্রয়োজন হয়। তখন আমরা GOTO নির্দেশ ব্যবহার করি। লক্ষ্য করলে দেখা যাবে GOTO নির্দেশে GO এবং TO শব্দ দুটো এক সঙ্গে গায়ে গায়ে লেখা—সাধারণ ইংরিজি ভাষার go to-এর মতো দুটো শব্দের মাঝে কোনও ফাঁক নেই। অনেক কম্পিউটারে GO এবং TO-এর মাঝে ভুল করে ফাঁক রাখলে প্রোগ্রাম চলবে না এবং কম্পিউটার জানিয়ে দেবে ভুল হয়েছে। কিন্তু বেসিকের কোনও কোনও আধুনিক সংস্করণে GOTO এবং GO TO দুটোই কম্পিউটার সঠিক নির্দেশ হিসেবে গ্রহণ করে। GOTO নির্দেশের বিশেষত্ব হচ্ছে এই নির্দেশ পেলে কম্পিউটার পরবর্তী নির্দেশে না গিয়ে তার নিয়ন্ত্রণ স্থানান্তরিত করবে অন্য একটি নির্দিষ্ট নির্দেশে—অর্থাৎ, তার ধারাবাহিক নির্দেশ পালনের নিয়ম ছেড়ে শর্তহীন ভাবে 'লাফ' দেবে অন্য একটি বিশেষ নির্দেশে। দু-একটি উদাহরণ দিলেই ব্যাপারটি স্পষ্ট হবে। যেমন,

```
 20  LET  X=10
┌30  GOTO 130
│40........
│....................
│....................
│120 ..........
└→130 PRINT X
 140 END
```

এই উদাহরণে 20 নম্বর নির্দেশ পালনের পরই কম্পিউটার পাচ্ছে 30 নম্বরের GOTO নির্দেশ বা 'লাফ' দেবার নির্দেশ। GOTO 130 মানে হচ্ছে 30 নম্বর নির্দেশের পর 40 নম্বর বা তার পরবর্তী অন্যান্য নির্দেশগুলোতে (... চিহ্নের সাহায্যে যা দেখানো হয়েছে) না গিয়ে সে তার নিয়ন্ত্রণ স্থানান্তরিত করবে সোজা 130 নম্বর নির্দেশে। এই ধরনের 'লাফ'কে বলা হয় শর্তহীন 'লাফ'। অর্থাৎ, কোনওরকম যুক্তি অথবা কারণ ছাড়াই কম্পিউটারের নিয়ন্ত্রণ 30 নম্বর নির্দেশের পর পৌঁছে যাবে 130 নম্বর নির্দেশে। কোনও নির্দেশে GOTO মূল শব্দের পরে অবশ্যই একটি বৈধ লাইন নম্বর বসাতে হবে। ওপরের উদাহরণে GOTO 130 না লিখে GOTO 30 লিখলে সেটা ভুল হত। কারণ 30 নম্বর নির্দেশ থেকে 30 নম্বর নির্দেশে যাওয়ার আদেশ অর্থহীন। আবার GOTO 135 লিখলে সেটাও ভুল হত। কারণ 135 নম্বর নামাঙ্কিত কোনও নিদেশ ওপরের প্রোগ্রামে নেই।

GOTO নির্দেশ বেসিক প্রোগ্রামে কীভাবে কাজে লাগে তা বোঝাতে 4 নম্বর প্রোগ্রামটি আমরা আবার ব্যবহার করব। ধরা যাক, একটি সমীকরণ দেওয়া আছে

$$y=1+x+2x^2$$

এখন x-এর মান যদি 1 থেকে 10 পর্যন্ত যে কোনও পূর্ণ সংখ্যা হয় তাহলে x-এর প্রতিটি মানের জন্য y-এর মান কত হবে? বোঝাই যাচ্ছে, x-এর দশটি মানের জন্য y-এরও দশটি মান বেরোবে। এই সমস্যাটির সমাধান করতে আমরা যদি 4-নম্বর প্রোগ্রামটি ব্যবহার করি তাহলে X-এর প্রতিটি মানের জন্য প্রোগ্রামটিকে একবার করে RUN করাতে হবে। অর্থাৎ, মোট দশ বার RUN লিখে প্রোগ্রামটি দশবার চালাতে হবে। কিন্তু একটি GOTO নির্দেশ ব্যবহার করে আমরা দশবার কী-বোর্ডের বোতাম টিপে RUN শব্দটি লেখার কষ্ট থেকে রেহাই পেতে পারি। ওই 4 নম্বর প্রোগ্রামে আমরা নিচের GOTO নির্দেশটি শুধু জুড়ে দেব, তাহলেই প্রোগ্রাম একবার RUN করালে X-এর বিভিন্ন মানের জন্য Y-এর বিভিন্ন মান ভি. ডি. ইউ.-তে ফুটে উঠবে। নির্দেশটি হল,

35 GOTO 10

এই নির্দেশটি যোগ করে 4 নম্বর প্রোগ্রামের চেহারা হবে এইরকম:

21

লুপ
```
10 INPUT X
20 LET Y=1+X+2*X↑2
30 PRINT Y
35 GOTO 10
40 END
```

এই প্রোগ্রামটি RUN করালে কম্পিউটার প্রথমেই X-এর মান জানতে

চাইবে—অর্থাৎ, ভি. ডি. ইউ.-তে ফুটে উঠবে জিজ্ঞাসা চিহ্ন (?)। তখন কী-বোর্ড থেকে X-এর প্রথম মান 1 যেই দেওয়া হবে অমনি ভি. ডি. ইউ.-তে ফুটে উঠবে Y-এর মান 4 এবং তার পরবর্তী লাইনে আবার একটি জিজ্ঞাসা চিহ্ন। অর্থাৎ, কম্পিউটার আবার X-এর মান জানতে চাইছে। এবারে যেই আমরা জানাব X-এর মান 2, সঙ্গে সঙ্গে ভি. ডি. ইউ.-তে ফুটে উঠবে Y-এর মান 11 এবং আর একটি জিজ্ঞাসা চিহ্ন। এইভাবে X-এর বহুরকম মানের জন্য Y-এর যথাযথ মান নির্ণয় করা যাবে এই প্রোগ্রামটি থেকে। ওপরের প্রোগ্রামটি লক্ষ্য করলেই দেখা যাবে, প্রথম X-এর মান কম্পিউটারকে জানানোর পর 30 নম্বর নির্দেশে এসে কম্পিউটার Y-এর মান 'ছাপাবে'। আর ঠিক তার পরেই 35 নম্বর নির্দেশ তার নিয়ন্ত্রণকে পাঠিয়ে দিচ্ছে প্রোগ্রামের শুরুতে 10 নম্বর নির্দেশে। ফলে তক্ষুণি আবার কম্পিউটার X-এর মান জানতে চাইছে। এইভাবে একটি GOTO নির্দেশ 10 নম্বর ও 35 নম্বর লাইন দুটির মধ্যে একটি চক্রাকার পথ বা অসীম লুপের সৃষ্টি করেছে। সুতরাং যত খুশি Y-এর মান নির্ণয় করার পর কম্পিউটারের অসীম লুপকে থামাতে হবে, নইলে কম্পিউটার ওই চক্রাকার পথে শুধু ঘুরপাক খেয়েই যাবে এবং একটা উত্তর জানিয়েই নতুন একটা জিজ্ঞাসা-চিহ্ন ফুটিয়ে তুলবে। অসীম লুপকে থামানোর জন্য কম্পিউটারের BREAK বোতাম টিপতে হয়।

এখানে বোঝানোর সুবিধের জন্য আমরা X-এর মান 1 থেকে 10 এর মধ্যে পূর্ণ সংখ্যা ধরেছি। তবে আদতে X-এর মান ধনাত্মক বা ঋণাত্মক যে-কোনও সংখ্যা হতে পারে। নিচে X ও Y-এর এরকম তিনটি মানের একটা সংক্ষিপ্ত তালিকা দেওয়া হল ঃ

X	Y
−0·000005	0·999995001
102·356	21056·85747
39	3082

খুব বড় সংখ্যা অথবা খুব ছোট সংখ্যাকে প্রকাশ করার সময়ে কম্পিউটার কখনও কখনও শর্টকাট পথ বেছে নেয়। যেমন, যদি কোনও উত্তর হয়

31592000000000

তাহলে কম্পিউটার লিখবে

3·1592E13

এখানে E অক্ষরটির অর্থ EXPONENT এবং E13 মানে হচ্ছে 13টা 10-কে পরস্পর গুণ। সুতরাং

$$3{\cdot}1592E13 = 3{\cdot}1592 \times 10^{13} = 31592000000000$$

ঠিক একইভাবে কোনও ছোট সংখ্যা বোঝাতেও E ব্যবহার করে কম্পিউটার। যেমন,

$$0{\cdot}0000000087621 = 8{\cdot}7621E-09$$

এখানে E – 09 এর অর্থ 9টা 10 দিয়ে পর পর ভাগ। অর্থাৎ, 8·7621 কে 9টা 10 দিয়ে ভাগ করলে তবেই 0·0000000087621 সংখ্যাটি পাওয়া যাবে। সুতরাং

$$0{\cdot}0000000087621 = 8{\cdot}7621E-09 = 8{\cdot}7621 \times 10^{-9}$$

$$= \frac{8{\cdot}7621}{10^9}$$

এবারে GOTO নির্দেশ ব্যবহার করে আরও কয়েকটি প্রোগ্রামের উদাহরণ দেওয়া হল ঃ

প্রথম উদাহরণে আমরা যে-কোনও সংখ্যার নামতা তৈরি করব। একটি সংখ্যাকে 1, 2, 3, 4,···ইত্যাদি সংখ্যা দিয়ে পরপর গুণ করে গুণফল যা হয় তাই হল সংখ্যাটির নামতা। এই নামতা আমরা নিচের বেসিক প্রোগ্রামটির সাহায্যে তৈরি করতে পারি ঃ

22

```
10  REM** MULTIPLICATION TABLE
20  LET A=O
30  INPUT "NUMBER" ; N
40  LET A=A+1
50  LET B=N*A
60  PRINT "MULTIPLE" ; B
70  PRINT "WANT MORE MULTIPLE ?",
80  PRINT "IF YES, WRITE Y," ;
90  PRINT "ELSE PRESS BREAK KEY—" ;
100 INPUT C$
110 GOTO 40
120 END
```

এই প্রোগ্রামটি RUN করালে কম্পিউটার N-সংখ্যাটি জানতে চাইবে। N-সংখ্যাটি তাকে জানানোর পর সে N-এর নামতা তৈরি করবে। ধরা যাক আমরা কী-বোর্ডের বোতাম টিপে N-এর মান দিলাম 27 ; তাহলে কম্পিউটার যে-উত্তর দেবে তা হল ঃ

```
RUN
MULTIPLE 27
WANT MORE MULTIPLE ?
IF YES, WRITE Y, ELSE PRESS BREAK KEY—?
```

যদি আমরা জানতে চাই 27 দ্বিগুণে কত, তাহলে উত্তরের শেষ লাইনের শেষে যে জিজ্ঞাসার চিহ্ন রয়েছে তার জবাবে কী-বোর্ড থেকে Y টাইপ করব। তাহলেই আমরা উত্তর পাব ঃ

```
MULTIPLE 54
WANT MORE MULTIPLE ?
IF YES, WRITE Y, ELSE PRESS BREAK KEY—?
```

এইভাবে একের পর এক Y টাইপ করে (এখানে ভুললে চলবে না, কী-বোর্ড থেকে প্রতিটি তথ্য বা মান টাইপ করার পর রিটার্ন বোতাম অবশ্যই টিপতে হবে) আমরা 27 এর বিভিন্ন গুণিতক পর পর পেয়ে যাব এবং তৈরি হয়ে যাবে 27 এর নামতা।

ওপরের প্রোগ্রামটিতে 100 নম্বর নির্দেশে যে সারি চলরাশি C$-এর কথা বলা আছে, তারই মান হিসেবে আমরা Y অক্ষরটি বার বার কম্পিউটারকে জানাচ্ছি এবং সে তৎক্ষণাৎ 110 নম্বর নির্দেশে যাচ্ছে, আর সেখান থেকে শর্তহীন 'লাফ' দিয়ে চলে যাচ্ছে 40 নম্বর নির্দেশে। 40 নম্বর নির্দেশে পৌঁছে সে A চলরাশিটির মান 1 বাড়িয়ে নিচ্ছে, এবং আবার বের করছে N ও A-র গুণফল। এইভাবে GOTO লুপের জন্য প্রোগ্রামটি বার বার A-র মান 1 করে বাড়াচ্ছে এবং N ও A-র গুণফল বের করছে।

দ্বিতীয় উদাহরণ হিসেবে আমরা আকাশবানী কলকাতা কেন্দ্রের C বিভাগের শ্রোতার শতকরা হার নির্ণয়ের সমস্যাটাই বেছে নেব (প্রোগ্রাম নং 19 দ্রষ্টব্য)। সকাল সাতটা থেকে রাত দশটা পর্যন্ত 15 ঘণ্টা সময়কে আমরা প্রথমে 1 থেকে 15 এই পনেরোটি ক্রমিক সংখ্যা দিয়ে চিহ্নিত করব। এই এক একটি সংখ্যাকে আমরা বলব টাইম স্লট নাম্বার। তাহলে নির্দিষ্ট কোনও ঘণ্টায় বা টাইম স্লট নাম্বারে C বিভাগের শ্রোতার শতকরা হার কত তা সহজেই বোঝা যাবে এবং তালিকার আকারে সেটা প্রকাশ করা যাবে।

23

```
Ok
LIST
10    PRINT
20    REM  CALCULATION OF  %  LISTENERS
30    REM  FOR CAL  C  OF  AIR
40    PRINT
50    PRINT "TIME SLOT NO.";
60    INPUT N
70    PRINT "TOTAL  NO. OF LISTENERS";
80    INPUT X
90    PRINT "NO. OF LISTENERS OF CAL  C ";
100   INPUT Y
110   LET Z = (Y/X)*100
120   PRINT "TIME SLOT NO.","
130   PRINT N,"        ";X           TOTAL LISTENERS"
140   PRINT "LISTENERS FOR CAL  C ";
150   PRINT "           % LISTENERS FOR CAL  C "
160   PRINT Y,"
170   GOTO  40                        ";Z;"%"
180   END
Ok
```

এই উদাহরণে GOTO নির্দেশ 40 ও 170 নম্বর লাইন দুটির মাঝে তৈরি করেছে একটি অসীম লুপ। পনেরোটি টাইম স্লট নাম্বারের হিসেব কষা হয়ে গেলে BREAK বোতাম টিপে অসীম লুপ থেকে বেরিয়ে আসা যাবে। ওপরের উদাহরণে আর একটি লক্ষ্যণীয় বিষয় হল বিভিন্ন PRINT নির্দেশে কমা ও সেমিকোলনের ব্যবহার। কম্পিউটারের 'ছাপানো' উত্তরকে সুন্দর ভাবে সাজানোর কাজে এই দুটি যতিচিহ্ন যথেষ্ট পারদর্শী। শুধু প্রোগ্রামার তাদের উপযুক্তভাবে ব্যবহার করতে পারলেই হল। ওপরের প্রোগ্রামটি RUN করালে কম্পিউটার যে-উত্তর দেবে তা পরের পৃষ্ঠায় দেখানো হল। এই প্রোগ্রামটি চালানো হয়েছে IDM-PC কম্পিউটারে।

```
Ok
RUN

TIME SLOT NO.? 1
TOTAL  NO. OF LISTENERS? 8893452
NO. OF LISTENERS OF CAL  C ? 903472
TIME SLOT NO.      TOTAL LISTENERS
 1                  8893452
LISTENERS FOR CAL  C          % LISTENERS FOR CAL  C
 903472                        10.15885 %

TIME SLOT NO.? 2
TOTAL  NO. OF LISTENERS? 3345298
NO. OF LISTENERS OF CAL  C ? 195867
TIME SLOT NO.      TOTAL LISTENERS
 2                  3345298
LISTENERS FOR CAL  C          % LISTENERS FOR CAL  C
 195867                        5.854994 %

TIME SLOT NO.?
```

GOTO নির্দেশ ব্যবহারের শেষ উদাহরণ হিসেবে 20 নম্বর প্রোগ্রামেরই নতুন একটি সংস্করণ নিচে দেওয়া হল। এই প্রোগ্রামটিও IDM-PC কম্পিউটারে চালানো হয়েছে।

24

```
Ok
LIST
10   PRINT
20   PRINT "HOW  OLD  ARE  YOY";
30   INPUT X
40   LET H = X*365*24
50   PRINT "DO  YOU  KNOW, YOU  HAVE"
60   PRINT "SLEPT  FOR";H/3;"HOURS!"
70   GOTO 10
80   END
Ok
RUN

HOW  OLD  ARE  YOY? 15
DO  YOU  KNOW, YOU  HAVE
SLEPT  FOR 43800 HOURS!

HOW  OLD  ARE  YOY? 40
DO  YOU  KNOW, YOU  HAVE
SLEPT  FOR 116800 HOURS!

HOW  OLD  ARE  YOY?
```

IF—THEN

বেসিক ভাষার এই নির্দেশটি অত্যন্ত শক্তিশালী এবং গুরুত্বপূর্ণ। GOTO নির্দেশে বিনাশর্তে কম্পিউটার তার নিয়ন্ত্রণকে স্থানান্তরিত করে। কিন্তু IF—THEN নির্দেশে কম্পিউটারের নিয়ন্ত্রণ নির্দিষ্ট একটি শর্তসাপেক্ষে বেসিক প্রোগ্রামের অন্য কোনও নির্দেশে 'লাফ' দেয়। কয়েকটা উদাহরণ দিলেই ব্যাপারটা পরিষ্কার হবে।

```
10  IF X=0 THEN 90
20  IF A$="BASIC" THEN 100
30  IF Y<=13 THEN 10
40  IF P>=4·2 THEN 70
```

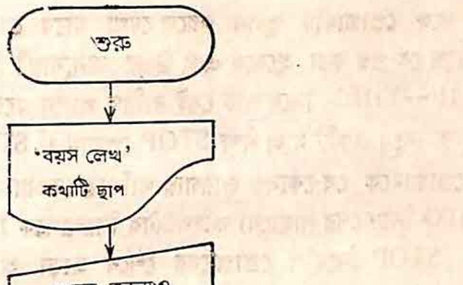
ওপরের চারটি উদাহরণেই দেখা যাচ্ছে, IF শব্দটির পরে লেখা শর্তটি যদি পালিত হয় তবেই কম্পিউটারের নিয়ন্ত্রণ স্থানান্তরিত হবে THEN-এর পরে যে-নির্দেশের নম্বর লেখা রয়েছে সেই নির্দেশে।

প্রথম উদাহরণে শর্ত হচ্ছে X=0; অতএব যদি (IF) X=0 হয় তাহলে (THEN) কম্পিউটার 90 নম্বর নির্দেশে নিয়ে যাবে তার নিয়ন্ত্রণকে এবং 90 নম্বর নির্দেশটি পালিত হবে। কিন্তু X-এর মান 0 ছাড়া অন্য কিছু হলে কম্পিউটার 90 নম্বর নির্দেশে না গিয়ে IF—THEN নির্দেশের ঠিক পরবর্তী নির্দেশটি পালন করবে।

ওপরের 20 নম্বর নির্দেশের উদাহরণে সারি চলরাশি A$-এর মান যদি BASIC-এর সমান হয় তাহলে পরবর্তী যে-নির্দেশ পালিত হবে তা হল 100 নম্বর নির্দেশ। একই ভাবে 30 নম্বর নির্দেশযুক্ত উদাহরণে শর্ত হচ্ছে Y<=13, অর্থাৎ, Y-এর মান যদি 13-এর চেয়ে ছোট অথবা সমান হয়। আর 40 নম্বরের শর্ত হল P যদি 4·2 এর চেয়ে বড় বা সমান হয়। এই সব শর্ত-সাপেক্ষেই কম্পিউটারের নিয়ন্ত্রণ অন্য কোনও নির্দিষ্ট নির্দেশে স্থানান্তরিত হবে।

এবারে আমরা IF—THEN নির্দেশ ব্যবহার করে একটা প্রোগ্রাম লিখব। আমাদের দেশে 21 বছর বয়েস না হলে ভোটাধিকার পাওয়া যায় না। আমরা এমন একটা প্রোগ্রাম লিখতে চাই যা কারও বয়েস বিচার করে জানাবে তার ভোটাধিকার আছে কি নেই।

সমস্যাটি সমাধানের জন্য আমরা একটা প্রবাহ-চিত্র আঁকব, এবং তারই নিচে IF—THEN নির্দেশ ব্যবহার করে তৈরি কম্পিউটার প্রোগ্রামটি দেওয়া হল ঃ



25

```
10 PRINT "TYPE YOUR AGE"
20 INPUT A
30 IF A>=21 THEN 60
40 PRINT "NOT ELIGIBLE TO VOTE"
50 STOP
60 PRINT "ELIGIBLE TO VOTE"
70 END

RUN
TYPE YOUR AGE
? 24
ELIGIBLE TO VOTE
```

প্রোগ্রামটিকে চালিয়ে INPUT হিসেবে 24 বছর বয়স জানালে কম্পিউটার কী উত্তর দেবে সেটা দেখানো হল।

প্রবাহ-চিত্রের সঙ্গে প্রোগ্রামটি তুলনা করলে দেখা যাবে যে প্রবাহ-চিত্রের বরফি চেহারার বাক্সে যে-প্রশ্ন করা হয়েছে এবং উত্তর অনুযায়ী সিদ্ধান্ত নেওয়া হয়েছে, প্রোগ্রামে IF—THEN নির্দেশটি সেই দায়িত্ব পালন করেছে। তাছাড়া এই প্রোগ্রামে আমরা নতুন একটি মূল শব্দ STOP পেলাম। STOP নির্দেশের সাহায্যে কোনও প্রোগ্রামকে যে-কোনও জায়গায় থামিয়ে দেওয়া যায়। অর্থাৎ, যেন কোনও GOTO নির্দেশের সাহায্যে কম্পিউটার নিয়ন্ত্রণকে END নির্দেশে পৌঁছে দেওয়া। STOP নির্দেশ প্রোগ্রামের শেষে ছাড়া আর যে-কোনও জায়গায় থাকতে পারে এবং একটি প্রোগ্রামে একাধিক STOP নির্দেশ থাকতে পারে। এইখানেই END নির্দেশের সঙ্গে STOP নির্দেশের তফাৎ। কারণ END নির্দেশ সব সময় থাকতে হবে বেসিক প্রোগ্রামের শেষে এবং একটি প্রোগ্রামে কখনওই একটির বেশি END নির্দেশ থাকতে পারবে না।

প্রতিটি নতুন নতুন লোকের বয়েসের জন্য ওপরের প্রোগ্রামটিকে নতুন করে RUN করাতে হবে। সেটা অসুবিধেজনক মনে হলে আমরা GOTO নির্দেশ ব্যবহার করে অসীম লুপের সৃষ্টি করতে পারি এবং STOP নির্দেশ-টিকেও ইচ্ছে হলে বাদ দিতে পারি। নিচে 25 নম্বর প্রোগ্রামের সেইরকম একটি রূপ দেখানো হল:

26

```
10 PRINT "TYPE YOUR AGE"
20 INPUT A
30 IF A>=21 THEN 60
40 PRINT "NOT ELIGIBLE TO VOTE"
50 GOTO 10
60 PRINT "ELIGIBLE TO VOTE"
70 GOTO 10
80 END
```

এই প্রোগ্রামটি RUN করিয়ে আগের উদাহরণের মতো A-র মান 24 জানালে কম্পিউটার কী উত্তর দেবে সেটা নিচে দেওয়া হল:

```
RUN
TYPE YOUR AGE
? 24
ELIGIBLE TO VOTE
TYPE YOUR AGE
?
```

এই প্রোগ্রামের অসীম লুপ থেকে বেরোতে হলে প্রয়োজন মতো BREAK বোতাম টিপতে হবে।

অনেক সময় IF—THEN নির্দেশ ব্যবহার করেও আমরা অসীম লুপের হাত থেকে রক্ষা পেতে পারি। উদাহরণ হিসেবে ধরা যাক, 1 থেকে 10 পর্যন্ত প্রত্যেকটি পূর্ণসংখ্যার বর্গ নির্ণয় করতে হবে। এই সমস্যার সমাধান করতে আমরা নিচের প্রোগ্রামটি লিখতে পারি ঃ

```
10  PRINT  1 * 1
20  PRINT  2 * 2
30  PRINT  3 * 3
.....................
... ..................
100  PRINT  10 * 10
```

কিন্তু দেখেই বোঝা যাচ্ছে, এই রকম প্রোগ্রাম লেখা যথেষ্ট অসুবিধেজনক। কারণ, 10 টি সংখ্যা না হয়ে যদি 1000 টি পূর্ণসংখ্যার বর্গ বের করতে হয় তাহলে 1000 টি PRINT নির্দেশ দিতে হবে। অতএব এই প্রোগ্রামটিরই সহজ ও সংক্ষিপ্ত রূপ হবে এইরকম ঃ

[27]

```
10  LET A=1
20  PRINT A * A;
30  LET A=A+1
40  IF A>10 THEN 60
50  GOTO 20
60  END
```

এখানে A-র মান 10-এর বেশি হলেই কম্পিউটার নিয়ন্ত্রণ 'লাফ' দিয়ে পেঁছে যাবে 60 নম্বর নির্দেশে। যদি আমরা 1 থেকে 1000 পর্যন্ত প্রতিটি পূর্ণসংখ্যার বর্গ নির্ণয় করতে চাই তাহলে 40 নম্বর নির্দেশে 10 এর জায়গায় 1000 লিখে দিলেই হল। তাহলে 1000 এর বর্গ বের করার পর প্রোগ্রাম শেষ হবে। অর্থাৎ, কম্পিউটার অসীম লুপের 'ফাঁদে' পড়বে না, আর প্রোগ্রামারকে BREAK বোতামও টিপতে হবে না।

ওপরের প্রোগ্রামটি RUN করালে উত্তর পাওয়া যাবে ঃ

```
RUN
 1  4  9  16  25  36  49  64
 81  100
```

IF—THEN নির্দেশের শেষ উদাহরণে আমরা 22 নম্বর প্রোগ্রামটি ব্যবহার করব। ধরা যাক, আমরা 1 থেকে 50 পর্যন্ত সংখ্যার নামতা বের করতে চাই। তাহলে 22 নম্বর প্রোগ্রামে N-এর সর্বোচ্চ মান হবে 50 এবং A-র সর্বোচ্চ মান স্বাভাবিক ভাবেই 10 ; অর্থাৎ, 50-এর নামতার শেষ গুণফলটি হবে

$$50 \times 10 = 500$$

সুতরাং প্রোগ্রামটিকে আমরা এইভাবে লিখব ঃ

28

```
10  REM .. MULTIPLICATION TABLE
20  INPUT "NUMBER" ; N
30  IF N>50 THEN 170
40  LET A=0
50  LET A=A+1
60  IF A>10 THEN 140
70  LET B=N * A
80  PRINT "MULTIPLE" ; B
90  PRINT "WANT MORE MULTIPLE ?",
100 PRINT "IF YES, WRITE Y," ;
110 PRINT "ELSE PRESS BREAK KEY—" ;
120 INPUT C$
130 GOTO 50
140 PRINT "THIS NUMBER-TABLE IS COMPLETE"
150 PRINT "NEXT NUMBER PLEASE"
160 GOTO 20
170 PRINT "SORRY ! THE NUMBER",
180 PRINT "IS GREATER THAN 50 !"
190 GOTO 20
200 END
```

প্রোগ্রামটি ব্যাখ্যা করার দায়িত্ব আমরা ছেড়ে দিলাম পাঠকের ওপরে। আর একই সঙ্গে তাকে এই প্রোগ্রামের একটা প্রবাহ-চিত্র আঁকতে অনুরোধ করছি।

IF—THEN—ELSE

কোনও কোনও কম্পিউটারে IF—THEN নির্দেশ ছাড়াও IF—THEN—ELSE নির্দেশের ব্যবস্থা আছে। IF—THEN নির্দেশে নির্দিষ্ট কোনও

শর্তসাপেক্ষে কম্পিউটারের নিয়ন্ত্রণ অন্য কোনও নির্দেশে 'লাফ' দেয়। অর্থাৎ, শর্তটি পূর্ণ হলে কম্পিউটারের নিয়ন্ত্রণ THEN মূল শব্দের পরে লেখা নির্দেশ-নম্বরে চলে যায়। আর শর্তটি পূর্ণ না হলে নিয়ন্ত্রণ চলে যায় IF—THEN নির্দেশটির ঠিক পরবর্তী নির্দেশে। এই প্রসঙ্গে 25 নম্বর প্রোগ্রামের উদাহরণে 30 ও 40 নম্বর নির্দেশ দুটি লক্ষ্য করা যাক ঃ

```
30  IF A>=21 THEN 60
40  PRINT "NOT ELIGIBLE TO VOTE"
```

যদি A-এর মান 21 এর সমান অথবা তার চেয়ে বড় হয়, তাহলে কম্পিউটার নিয়ন্ত্রণ স্থানান্তরিত হবে 60 নম্বর নির্দেশে। আর অন্যথায়, A যদি 21 এর চেয়ে ছোট হয়, তাহলে নিয়ন্ত্রণ চলে যাবে 30 এর পরবর্তী 40 নম্বর নির্দেশে।

ওপরের 30 নম্বর নির্দেশকে আমরা এভাবেও লিখতে পারতাম ঃ

```
30  IF A>=21 THEN 60 ELSE 40
```

অর্থাৎ, কম্পিউটারকে যেন আমরা বলছি, 'শর্ত পূর্ণ হলে 60 নম্বর নির্দেশে যাও, আর পূর্ণ না হলে 40 নম্বর নির্দেশে যাও।'

সুতরাং এই আলোচনা থেকেই IF—THEN—ELSE নির্দেশের তাৎপর্য বোঝা যাচ্ছে। অর্থাৎ, 'IF শব্দের পরে লেখা শর্ত (এক্ষেত্রে A >=21) পূর্ণ হলে THEN-এর পরে লেখা নির্দেশ-নম্বরে (এক্ষেত্রে 60) যাও, আর শর্ত পূর্ণ না হলে ELSE-এর পরে লেখা নির্দেশ নম্বরে (এক্ষেত্রে 40) যাও।'

তবে ওপরের উদাহরণে 'ELSE 40' কথা দুটি নেহাৎ অপ্রয়োজনেই লেখা হয়েছে। কারণ এই প্রোগ্রামের জন্য IF—THEN নির্দেশই যথেষ্ট, IF—THEN—ELSE নির্দেশের প্রয়োজন নেই।

এবারে IF—THEN—ELSE নির্দেশ ব্যবহার করে আমরা একটি প্রশ্নের উত্তর বের করব।

প্রশ্নটি হল এই ঃ একটি ত্রিভুজের তিনটি বাহু a, b ও c দেওয়া আছে। কম্পিউটার প্রোগ্রামের সাহায্যে বের কর ত্রিভুজটি সমবাহু, সমদ্বিবাহু না বিষমবাহু।

এই সমস্যাটি সমাধানের প্রবাহ-চিত্র আমরা আগেই দিয়েছি (81 পৃষ্ঠায়)। সেই প্রবাহ-চিত্র অনুসরণ করে সমাধানের বেসিক প্রোগ্রামটি নিচে লেখা হল ঃ

29

```
10  REM .. TYPE OF A TRIANGLE
20  PRINT "TYPE OF A TRIANGLE"
30  PRINT "SIDE A=";
40  INPUT A
```

```
50  PRINT "SIDE B=" ;
60  INPUT B
70  PRINT "SIDE C=" ;
80  INPUT C
90  IF A=B THEN IF B=C THEN 140 ELSE 160
100 IF B=C THEN 160
110 IF A=C THEN 160
120 PRINT "SCALENE TRIANGLE"
130 GOTO 170
140 PRINT "EQ. TRIANGLE"
150 GOTO 170
160 PRINT "ISO. TRIANGLE"
170 GOTO 20
180 END
```

ওপরের প্রোগ্রামটি RUN করালে কম্পিউটারের ভি. ডি. ইউ.-তে ফুটে উঠবে :

```
RUN
TYPE OF A TRIANGLE
SIDE A=? 12
SIDE B=? 14
SIDE C=? 12
ISO. TRIANGLE
TYPE OF A TRIANGLE
SIDE A=?
```

ওপরের প্রোগ্রামটি ত্রিভুজের তিনটে বাহু—যথাক্রমে a, b ও c—পরস্পর তুলনা করছে। তুলনা করার কাজটা শুরু হচ্ছে 90 নম্বর নির্দেশ থেকে। সেখানে IF—THEN—ELSE নির্দেশের সাহায্যে কম্পিউটার প্রথমে a ও b কে তুলনা করছে। যদি a ও b সমান হয় তাহলে কম্পিউটার তখন তুলনা করে দেখবে b ও c পরস্পর সমান কি না। যদি b ও c সমান হয় তাহলে ত্রিভুজের তিনটে বাহুই পরস্পর সমান হল এবং কম্পিউটার নিয়ন্ত্রণ তখন চলে যাবে 140 নম্বর নির্দেশে। সেখানে পৌঁছে লিখে দেবে EQ. TRIANGLE (অর্থাৎ, সমবাহু ত্রিভুজ)। তারপর পরবর্তী নির্দেশে গিয়ে সেই নির্দেশ অনুযায়ী নিয়ন্ত্রণ স্থানান্তরিত হবে 170 নম্বর নির্দেশে এবং সেখান থেকে আবার 20 নম্বর লাইনে।

90 নম্বর নির্দেশে যদি a ও b পরস্পর সমান না হয় তাহলে সঙ্গে সঙ্গেই নিয়ন্ত্রণ চলে যাবে পরের লাইন 100 নম্বরে। সেখানে গিয়ে তুলনা করবে b ও c-এর মধ্যে। b ও c সমান হলে নিয়ন্ত্রণ 160 নম্বর লাইনে গিয়ে লিখে দেবে ISO. TRIANGLE (অর্থাৎ, সমদ্বিবাহু ত্রিভুজ), আর অন্যথায় চলে যাবে 110 নম্বর নির্দেশে। সেখানে a ও c-এর মধ্যে তুলনা করা হবে। যদি a ও c অসমান হয় তাহলে 120 নম্বর লাইনে গিয়ে কম্পিউটার লিখে দেবে SCALENE TRIANGLE (অর্থাৎ, বিষমবাহু ত্রিভুজ)। তারপর 130 ও 170 নম্বর নির্দেশ হয়ে নিয়ন্ত্রণ ফিরে যাবে 20 নম্বর নির্দেশে।

90 নম্বর লাইনে ধরা যাক, a=b কিন্তু b≠c। তাহলে কম্পিউটার নিয়ন্ত্রণ ELSE-এর পরবর্তী উল্লেখ অনুযায়ী 160 নম্বর নির্দেশে চলে যাবে এবং লিখে দেবে ISO. TRIANGLE (অর্থাৎ, সমদ্বিবাহু ত্রিভুজ)।

এইভাবে সবরকম সম্ভাব্য যুক্তি বিচার করে তবেই কম্পিউটার তার সিদ্ধান্ত জানাবে।

ইচ্ছে করলে 29 নম্বর প্রোগ্রামটিকে অনেকভাবে উন্নত করা যায়। যেমন,

১। আমরা জানি যে, ত্রিভুজের কোনও বাহুর মাপ কখনও ঋণাত্মক হতে পারে না।

২। আমরা জানি যে, ত্রিভুজের কোনও দুটি বাহুর যোগফল তৃতীয় বাহু অপেক্ষা বৃহত্তর। বাহুগুলোর মাপ এই শর্ত পূরণ না করলে তা দিয়ে কোনও ত্রিভুজ আঁকা সম্ভব নয়।

29 নম্বর প্রোগ্রামে এই শর্তগুলো মানার জন্য কোনও ব্যবস্থা নেওয়া হয়নি। এবারে আমরা ওপরের দুটি শর্ত ব্যবহার করে 29 নম্বর প্রোগ্রামটিকে আরও উন্নত এবং নির্ভুল রূপ দেব। প্রোগ্রামটির ব্যাখ্যা আমরা ছেড়ে দিলাম পাঠকের ওপরে এবং এর একটি প্রবাহ-চিত্র আঁকার জন্য তাকে অনুরোধ জানাচ্ছি।

30

```
10  REM ·· TYPE OF A TRIANGLE
20  PRINT "TYPE OF A TRIANGLE"
30  PRINT "SIDE A=";
40  INPUT A
50  IF A<=0 THEN 240
60  PRINT "SIDE B=";
70  INPUT B
80  IF B<=0 THEN 240
```

```
90  PRINT "SIDE C=";
100 INPUT C
110 IF C <=0 THEN 240
120 IF (A+B)<=C THEN 240
130 IF (B+C)<=A THEN 240
140 IF (C+A)<=B THEN 240
150 IF A=B THEN IF B=C THEN 200 ELSE 220
160 IF B=C THEN 220
170 IF C=A THEN 220
180 PRINT "SCALENE TRIANGLE"
190 GOTO 250
200 PRINT "EQ. TRIANGLE"
210 GOTO 250
220 PRINT "ISO. TRIANGLE"
230 GOTO 250
240 PRINT "SORRY ! NO TRIANGLE POSSIBLE !"
250 GOTO 20
260 END
```

ইচ্ছে করলে এই প্রোগ্রামটিকে আরও সংক্ষিপ্ত করা যায়। তবে প্রথম দিকে প্রোগ্রাম সংক্ষেপ করার চেষ্টার চেয়ে নির্ভুল প্রোগ্রাম লেখার চেষ্টাই করা উচিত।

FOR—TO—STEP—NEXT

বেসিক ভাষায় IF—THEN নির্দেশের মতোই আর একটি প্রয়োজনীয় এবং উপকারী নির্দেশ হল FOR—TO—STEP—NEXT নির্দেশ। এটাকে অনেক সময় সংক্ষেপে FOR-NEXT নির্দেশ বা লুপ নামে অভিহিত করা হয়। এর আগের কয়েকটি প্রোগ্রামে আমরা দেখেছি (21, 22, 23, 24 ও 26 নম্বর প্রোগ্রাম) GOTO নির্দেশ ব্যবহার করে প্রোগ্রামে যে-অসীম লুপের সৃষ্টি হয় তা থেকে বেরিয়ে আসতে হলে আমাদের BREAK বোতাম টিপতে হয়, অথবা GOTO-এর সঙ্গে উপযুক্ত ভাবে একটা IF—THEN নির্দেশ ব্যবহার করতে হয় (যেমন 27 নম্বর প্রোগ্রাম)। একটা প্রোগ্রামকে লুপে কতবার পাক খাওয়াতে হবে সেই সংখ্যাটা জানা না থাকলে IF—THEN কিংবা GOTO নির্দেশ ব্যবহার করাটাই ঠিক। কিন্তু বেশিরভাগ সময়েই লুপে পাক খাওয়ানোর সংখ্যাটা আমাদের জানা থাকে। আর তখন FOR—TO—STEP

—NEXT নির্দেশ ব্যবহার করে খুব সহজেই আমরা প্রোগ্রামটা তৈরি করতে পারি।

FOR—TO—STEP—NEXT নির্দেশকে আমরা কয়েকটা ভাগে ভেঙে নিয়ে ধাপে ধাপে ব্যাখ্যা করব।

প্রথমে আমরা আলোচনা করব FOR—TO নির্দেশটুকু। নিচে FOR—TO নির্দেশের তিনটি উদাহরণ দেওয়া হল ঃ

```
10  FOR X=1 TO 100
20  FOR P1=2·7 TO 13·7
30  FOR X2=−4·3 TO 12·7
```

এই উদাহরণগুলোতে X, P1, X2 হল সাংখ্যিক চলরাশি। এই চলরাশির মান নির্দিষ্ট নয়, বরং পরিবর্তনশীল। যেমন10 নম্বর নির্দেশে X-এর মান 1 থেকে শুরু হয়ে শেষ হবে 100 তে পেঁছে। FOR—TO নির্দেশ দিয়ে কোনও প্রোগ্রামে একটি লুপের শুরু হয়। 10 নম্বর নির্দেশ দিয়ে যে-লুপের শুরু হবে তাতে প্রথম বারে X-এর মান হবে 1। তারপর যতবারই ঘুরে ঘুরে লুপটি বার বার সম্পন্ন হবে ততবারই X-এর মান 1 করে বাড়বে। অর্থাৎ, প্রথম বারে লুপ পরিক্রমা করে কম্পিউটারের নিয়ন্ত্রণ যখন আবার 10 নম্বর নির্দেশে ফিরে আসবে তখন X-এর মান হবে 2। তৃতীয় বারে 3। এইভাবে প্রতিটি পাকে X-এর মান 1 করে বাড়তে থাকবে এবং X-এর মান 100 পর্যন্ত পেঁছে লুপটি শেষ হবে নিজে থেকেই—তখন আর BREAK বোতাম টেপার দরকার হবে না।

FOR—TO নির্দেশে যে-চলরাশি ব্যবহার করা হয় তার মান সব সময় 1 করে বাড়ে। এ থেকেই আমরা বুঝতে পারছি, 10 নম্বর নির্দেশযুক্ত লুপটি কম্পিউটার 100 বার সম্পন্ন করবে।

একই নিয়ম অনুযায়ী 20 নম্বর নির্দেশে P1 চলরাশির মান যথাক্রমে হবে

$$2{\cdot}7,\ 3{\cdot}7,\ 4{\cdot}7,\ \ldots,\ 13{\cdot}7$$

অর্থাৎ, লুপটি সম্পন্ন হবে মোট 12 বার। আর 30 নম্বর নির্দেশের FOR—TO লুপটি সম্পন্ন হবে মোট 18 বার।

একটি FOR—TO লুপ কতবার সম্পন্ন হবে তা হিসেব করার উপায় হল, চলরাশির সর্বনিম্ন ও সর্বোচ্চ মানের বিয়োগফলের সঙ্গে 1 যোগ করা। যেমন, 30 নম্বর নির্দেশে লুপে মোট আবর্তনের সংখ্যা হল ঃ

$$[\,12{\cdot}7-(\,-4{\cdot}3\,)\,]+1=18$$

এবারে আমরা FOR—TO নির্দেশকে ভিত্তি করে FOR—TO—STEP নির্দেশ আলোচনা করব।

FOR—TO নির্দেশে চলরাশির মান লুপের প্রতিবার আবর্তনের সঙ্গে সঙ্গে 1 করে বাড়ে। যদি আমরা চাই যে চলরাশিটির মান 1 এর পরিবর্তে অন্য কোনও হারে বেড়ে উঠুক, তাহলে আমাদের ব্যবহার করতে হবে FOR—TO—STEP নির্দেশ। আর চলরাশির মানের বৃদ্ধির যে-হার আমরা চাই, তা লিখতে হবে STEP মূল শব্দটির পরে। যেমন,

```
10  FOR X=1 TO 100 STEP 0·5
20  FOR P1=12·7 TO 0·7 STEP—1·0
30  FOR X2=—4·3 TO 12·7 STEP 0·01
40  FOR A=1 TO 97 STEP 3
```

ওপরের 20 নম্বর নির্দেশে P1 চলরাশির মানগুলো হবে এইরকম ঃ

12·7, 11·7, 10·7, 9·7···, 1·7, 0·7

আবার 40 নম্বর নির্দেশে A চলরাশির মানগুলো হবে,

1, 4, 7, 10, 13, ······, 91, 94, 97

সুতরাং চলরাশির মানের পরিবর্তনের হার (STEP) ধনাত্মক, ঋণাত্মক, দশমিক যা-কিছু হতে পারে। এছাড়াও, চলরাশির সর্বোচ্চ-সর্বনিম্ন মান এবং মান পরিবর্তনের হার (STEP) অন্য কোনও চলরাশি অথবা ফর্মুলা বা সূত্র হতে পারে। যেমন,

```
10  FOR Q=X1 TO X2 STEP X3
20  FOR P=(R+S)/3 TO T ↑ 3 STEP Q+2
```

বেসিকের কোনও কোনও সংস্করণে STEP শব্দটির বদলে BY ব্যবহার করা হয়।

আমরা দেখলাম যে প্রোগ্রামে কোনও লুপ শুরু হয় FOR—TO—STEP নির্দেশ দিয়ে। কিন্তু লুপটি শেষ হয় কোথায়? অর্থাৎ, চলরাশির মান সর্বোচ্চ বা সর্বনিম্ন মানে না পৌঁছনো পর্যন্ত কোন্ নির্দেশ কম্পিউটার নিয়ন্ত্রণকে বার-বার ফেরত পাঠাবে FOR—TO—STEP নির্দেশে? এই গুরুত্বপূর্ণ কাজটি করে NEXT মূল শব্দটি। FOR—TO—STEP দিয়ে যে-লুপ শুরু হয়, তা শেষ হয় NEXT দিয়ে। সুতরাং সব মিলিয়ে লুপ সংক্রান্ত নির্দেশটি দাঁড়াল FOR—TO—STEP—NEXT। FOR—TO—STEP ও NEXT নির্দেশ দুটির মাঝে অন্তত একটি অন্য কোনও নির্দেশ থাকতেই হবে। দু-একটি উদাহরণ দিলেই ব্যাপারটা পরিষ্কার হয়ে যাবে।

```
    ┌→40  FOR A=1 TO 99 STEP 2
    │      ......
লুপ ↑      ......
    │      ......
    └←130 NEXT A
```

ওপরের উদাহরণে 40 ও 130 নম্বর নির্দেশের মধ্যে একটি লুপ তৈরি হয়েছে। A চলরাশিটির প্রতিটি মানের জন্য (অর্থাৎ, 1, 3, 5, ..., 95, 97, 99) কম্পিউটার লুপটি একবার করে সম্পন্ন করবে। সুতরাং মোট 50 বার লুপটি সম্পন্ন হবে। প্রথম 40 নম্বর নির্দেশে পেঁছে কম্পিউটার A চলরাশির মান 1 ধরে নেবে এবং 40 এর পরবর্তী নির্দেশগুলো পালন করতে করতে 130 নম্বর নির্দেশে পেঁছবে। সেখানে 'NEXT A' নির্দেশটির অর্থ, নিয়ন্ত্রণ আবার ফিরে যাক FOR—TO—STEP নির্দেশে—অর্থাৎ, 40 নম্বর নির্দেশে—এবং A-এর বর্ধিত মান 3 নিয়ে লুপটি দ্বিতীয়বার সম্পন্ন করুক। এইভাবে A-এর মান 99 এ না পেঁছনো পর্যন্ত 130 নম্বরের 'NEXT A' নির্দেশটি বার বার নিয়ন্ত্রণকে ফেরত পাঠাবে 40 নম্বর নির্দেশে।

এবারে আমরা FOR—NEXT নির্দেশের ব্যবহার দেখার জন্য কয়েকটা বেসিক প্রোগ্রাম লিখব।

প্রথমে আমরা 27 নম্বর প্রোগ্রামটিকে FOR-NEXT নির্দেশ ব্যবহার করে নতুনভাবে লিখব।

31

```
    ┌→10  FOR A=1 TO 10
লুপ ↑ 20  PRINT A*A
    └←30  NEXT A
      40  END
```

27 ও 31 নম্বর প্রোগ্রাম দুটিকে তুলনা করে আমরা দেখতে পাচ্ছি FOR—NEXT নির্দেশ ব্যবহার করে 31 নম্বর প্রোগ্রামটি আরও সংক্ষেপে লেখা সম্ভব হয়েছে। এই প্রোগ্রামটি RUN করালে আমরা 27 নম্বর প্রোগ্রামের মতো একই উত্তর পাব।

নিচে FOR—NEXT লুপের আরও কয়েকটি উদাহরণ দেওয়া হল ঃ

32

```
10  FOR A=1 TO 8
20  PRINT A, 2*A, 3*A
30  1F A=4 THEN 50
40  GOTO 60
```

```
50  PRINT "HALFWAY THROUGH"
60  NEXT A
70  END
```

এই প্রোগ্রামটি RUN করালে উত্তর পাব এই রকম ঃ

```
RUN
1            2            3
2            4            6
3            6            9
4            8            12
HALFWAY      THROUGH
5            10           15
6            12           18
7            14           21
8            16           24
```

33

```
10  LET X=0
20  FOR I=1 TO 100
30  LET X=X+I
40  IF X >=1000 THEN 60
50  NEXT I
60  PRINT I ; "ঃ" ; X
70  END
```

এই প্রোগ্রামটিতে (1+2+3+4+ ···) এই শ্রেণীটির যোগফল পাওয়া যাবে। যোগফল যখনই 1000 এর সমান অথবা বেশি হবে তখনই প্রোগ্রামটির লুপে ঘোরা শেষ হবে। অর্থাৎ, শর্তসাপেক্ষে লুপ থেকে বেরিয়ে আসার কাজটি সম্পন্ন করছে 40 নম্বর নির্দেশ। এইভাবে কোনও শর্তের ওপরে নির্ভর করে FOR—NEXT লুপ থেকে বেরিয়ে আসা যায়, কিন্তু বাইরে থেকে FOR—NEXT লুপে ঢোকা যায় না। এই প্রোগ্রামটি RUN করালে উত্তর পাওয়া যাবে ঃ

```
RUN
45 ঃ 1035
```

অর্থাৎ, 1 থেকে 45 পর্যন্ত পূর্ণসংখ্যাগুলো যোগ করলে যোগফল 1035 হবে। যেহেতু যোগফল 1000 পেরিয়ে যাবে সেহেতু 40 নম্বর নির্দেশ অনুযায়ী নিয়ন্ত্রণ স্থানান্তরিত হবে 60 নম্বর নির্দেশে এবং প্রোগ্রামটি শেষ হবে।

এবারের উদাহরণে আমরা কোনও পূর্ণসংখ্যার ফ্যাক্টোরিয়াল বা ক্রমগুণিত বের করব। যদি কোনও পূর্ণসংখ্যা n হয়, তাহলে 'ফ্যাক্টোরিয়াল n' কে প্রকাশ করা হয় এইভাবে : $\lfloor n$ অথবা n!। 'ফ্যাক্টোরিয়াল n'-এর অর্থ হল :

$$n! = 1 \times 2 \times 3 \times \cdots \times n$$

অর্থাৎ, 1 থেকে শুরু করে n পূর্ণসংখ্যাটি পর্যন্ত সমস্ত পূর্ণসংখ্যাগুলোর গুণফল। সুতরাং, $10! = 1 \times 2 \times 3 \times \cdots \times 10 = 3628800$

নিচের বেসিক প্রোগ্রামটির সাহায্যে আমরা যে-কোনও পূর্ণসংখ্যার ক্রমগুণিত বের করতে পারব।

34

```
10   REM ·· FACTORIAL OF A NUMBER
20   PRINT "N=";
30   INPUT N
40   LET F=1
50   FOR K=1 TO N
60   LET F=F*K
70   NEXT K
80   PRINT "N="; (K-1), "N!="; F
90   GOTO 20
100  END
```

এই প্রোগ্রামটি RUN করালে উত্তর পাওয়া যাবে :

```
RUN
N=? 10
N=10
N!=3628800
N=? 45
N=45
N!=1·196222209E56
N=?
```

ওপরের উত্তরে কী-বোর্ড থেকে জানানো তথ্যকে তলায় দাগ টেনে দেখানো হল (10 ও 45)। লক্ষ্য করার বিষয়, 80 নম্বর নির্দেশে (K-1)-এর মান N-এর মান হিসেবে 'ছাপতে' বলা হয়েছে। এর অর্থ, 50 নম্বর নির্দেশে K যখন তার সর্বোচ্চ মান N-এ পেঁছিবে তখন লুপের ভেতরের সমস্ত নির্দেশ (এক্ষেত্রে শুধু 60 নম্বর নির্দেশ) পালন করে কম্পিউটার নিয়ন্ত্রণ 70 নম্বর নির্দেশে পেঁছে 'NEXT K' নির্দেশ অনুযায়ী K-এর মান আবার 1 বাড়িয়ে দেবে। অর্থাৎ,

তখন K=N+1। এই অবস্থায় সে আবার ফিরে যাবে 50 নম্বর নির্দেশে। কিন্তু সেখানে গিয়ে সে দেখবে K-এর মান N-এর মান ছাড়িয়ে গেছে। সুতরাং তৎক্ষণাৎ সে 'লাফ' দিয়ে চলে যাবে লুপের বাইরে 80 নম্বর নির্দেশে। তখন যেহেতু K=N+1, সেহেতু উত্তর 'ছাপানোর' সময়ে আমাদের N=K−1 হিসেব করতে হবে। তবেই N-এর সঠিক মান ফুটে উঠবে ভি. ডি. ইউ.-তে।

এবারে আমরা শেষ উদাহরণ হিসেবে একটা প্রতিযোগিতামূলক পরীক্ষার সর্বোচ্চ নম্বর নির্ণয়ের সমস্যা বেছে নেব। ধরা যাক, একটি পরীক্ষায় 250 জন ছাত্র পরীক্ষা দিয়েছে। সেই পরীক্ষায় সবচেয়ে বেশি নম্বর কত উঠেছে তা বের করতে হবে। এর বেসিক প্রোগ্রামটি হবে এই রকম ঃ

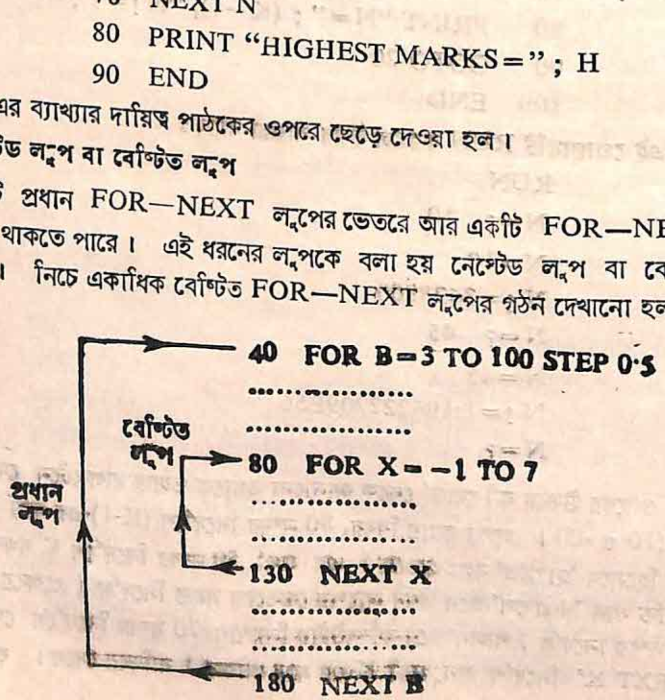
35

```
10 LET H=0
20 FOR N=1 TO 250
30 PRINT "MARKS=";
40 INPUT M
50 IF M<=HTHEN 70
60 LET H=M
70 NEXT N
80 PRINT "HIGHEST MARKS="; H
90 END
```

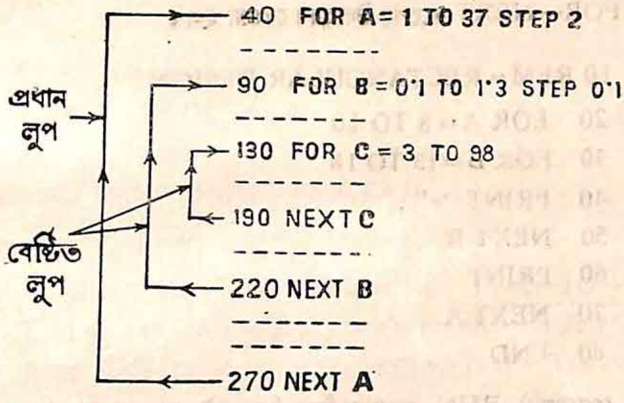
এর ব্যাখ্যার দায়িত্ব পাঠকের ওপরে ছেড়ে দেওয়া হল।

নেস্টেড লুপ বা বেষ্টিত লুপ

একটি প্রধান FOR—NEXT লুপের ভেতরে আর একটি FOR—NEXT লুপ থাকতে পারে। এই ধরনের লুপকে বলা হয় নেস্টেড লুপ বা বেষ্টিত লুপ। নিচে একাধিক বেষ্টিত FOR—NEXT লুপের গঠন দেখানো হল ঃ



একটি প্রধান লুপের ভেতরে একাধিক বেষ্টিত লুপ থাকতে পারে। যেমন,

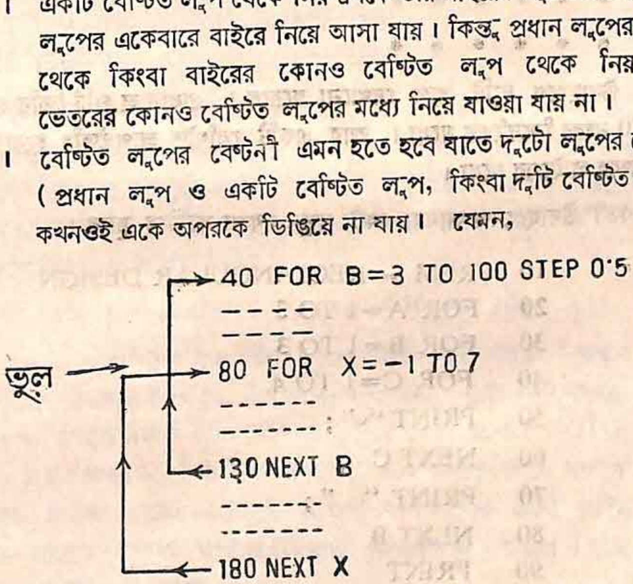


বেষ্টিত লুপ সম্পর্কে নিচের নিয়মগুলো সবসময় মনে রাখতে হবে :

১। প্রতিটি লুপের চলরাশির নাম বিভিন্ন হতে হবে। যেমন, ওপরের উদাহরণে A, B ও C।

২। একটি বেষ্টিত লুপ থেকে নিয়ন্ত্রণকে তার বাইরের লুপে অথবা প্রধান লুপের একেবারে বাইরে নিয়ে আসা যায়। কিন্তু প্রধান লুপের বাইরে থেকে কিংবা বাইরের কোনও বেষ্টিত লুপ থেকে নিয়ন্ত্রণকে ভেতরের কোনও বেষ্টিত লুপের মধ্যে নিয়ে যাওয়া যায় না।

৩। বেষ্টিত লুপের বেষ্টনী এমন হতে হবে যাতে দুটো লুপের বেষ্টনী (প্রধান লুপ ও একটি বেষ্টিত লুপ, কিংবা দুটি বেষ্টিত লুপ) কখনওই একে অপরকে ডিঙিয়ে না যায়। যেমন,



এই উদাহরণটিতে দুটি লুপের বেষ্টনী একে অপরকে 'ছেদ' করেছে। ফলে

কম্পিউটার জানিয়ে দেবে প্রোগ্রাম ভুল হয়েছে। এবারে বেষ্টিত লুপ সম্পন্ন কয়েকটি FOR—NEXT লুপের উদাহরণ দেওয়া হল ঃ

36

```
10 REM •• RECTANGULAR DESIGN
20  EOR A=8 TO 10
30  FOR B=13 TO 18
40  PRINT "*" ;
50  NEXT B
60  PRINT
70  NEXT A
80  END
```

এই প্রোগ্রামটি RUN করালে ভি. ডি. ইউ.-তে ফুটে উঠবে নিচের নকশাটি ঃ

```
RUN
*  *  *  *  *  *
*  *  *  *  *  *
*  *  *  *  *  *
```

এই উদাহরণে দুটি লুপ দেখানো হয়েছে। প্রধান লুপটি তৈরি হয়েছে 20 ও 70 নম্বর নির্দেশের মধ্যে। আর একটি বেষ্টিত লুপ তৈরি হয়েছে 30 ও 50 নম্বর লাইনের মধ্যে।

পরবর্তী উদাহরণেও আমরা একটা নতুন নকশা ফুটিয়ে তুলব।

37

```
10   REM •• RECTANGULAR DESIGN
20   FOR A=1 TO 3
30   FOR B=1 TO 3
40   FOR C=1 TO 4
50   PRINT "*" ;
60   NEXT C
70   PRINT "  " ;
80   NEXT B
90   PRINT
100  NEXT A
110  END
```

এই প্রোগ্রামটি RUN করালে আমরা নিচের নকশাটি দেখতে পাব :

```
RUN
* * * *    * * * *    * * * *
* * * *    * * * *    * * * *

* * * *    * * * *    * * * *
```

এই প্রোগ্রামে মোট তিনটি ল্ুপ ব্যবহার করা হয়েছে।

ON—GOTO

GOTO, IF—THEN, IF—THEN—ELSE ও FOR—TO—STEP—NEXT এর পর আর একটি শক্তিশালী নির্দেশ হল ON—GOTO।

ধরা যাক কোনও একটি বেসিক প্রোগ্রামে নিচের পাঁচটি IF—THEN নির্দেশ পরপর ব্যবহার করার দরকার হয়ে পড়েছে :

```
120  ..................
130  IF  K=1  THEN  90
140  IF  K=2  THEN 260
150  IF  K=3  THEN 230
160  IF  K=4  THEN  90
170  IF  K=5  THEN 300
```

এর অর্থ হল 120 নম্বর নির্দেশের পর প্রোগ্রামটি পাঁচটি শর্ত অনুযায়ী কম্পিউটার নিয়ন্ত্রণকে পাঁচটি বিভিন্ন নির্দেশে নিয়ে যাবে। প্রোগ্রামে এই ধরনের বহুমুখী শাখাপথ থাকলে অনেকগুলো IF—THEN নির্দেশ ব্যবহার করা বেশ অসুবিধেজনক। তাই এগুলোকে সংক্ষিপ্তভাবে মাত্র একটি ON—GOTO নির্দেশ দিয়ে প্রকাশ করা যায় :

```
130  ON K GOTO 90, 260, 230, 90, 300
```

এর অর্থ হল K চলরাশিটির মান যদি 1 হয় তাহলে কম্পিউটার নিয়ন্ত্রণ স্থানান্তরিত হবে 90 নম্বর নির্দেশে, 2 হলে 260 নম্বরে, 3 হলে 230 নম্বরে, 4 হলে আবার 90 নম্বরে এবং 5 হলে 300 নম্বর নির্দেশে। এক্ষেত্রে GOTO-এর পরে লেখা লাইন নম্বর বা নির্দেশ-নম্বরগুলো একে অপরের থেকে সম্পূর্ণ ভিন্ন হতে পারে, অথবা তাদের মধ্যে দুই বা তার বেশি সংখ্যক একই লাইন নম্বর থাকতে পারে। যেমন, ওপরের উদাহরণে K-এর মান 1 অথবা 4 হলে নিয়ন্ত্রণ চলে যাচ্ছে 90 নম্বর লাইনে।

কোনও ON—GOTO নির্দেশে K-এর মান যদি ভগ্নাংশ বা দশমিক হয় তাহলে কম্পিউটার নিজে থেকেই শুধু K-এর পূর্ণসংখ্যার অংশটুকু গ্রহণ

করবে এবং দশমিক অংশটুকু বর্জন করবে। অর্থাৎ, যদি K = 4·68 হয় তাহলে কম্পিউটার ·68 অংশটুকু বাদ দিয়ে ধরে নেবে, K = 4, এবং সেই অনুযায়ী নিয়ন্ত্রণকে স্থানান্তরিত করবে।

ON—GOTO নির্দেশে K-এর জায়গায় কোনও ফর্মুলা বা ব্যঞ্জক থাকতে পারে। যেমন,

130 ON 3*X+Y ↑ 3·5 GOTO 90, 260, 230, 90, 300

এখানে $(3x + y^{3 \cdot 5})$ এই ব্যঞ্জকটির মানের ওপরে নির্ভর করে নিয়ন্ত্রণ স্থানান্তরিত হবে।

বেসিক ভাষার কোনও কোনও সংস্করণে ON—GOTO-এর বদলে ON—THEN ব্যবহার করা যায়। তবে এর জন্য নির্দেশটির নিয়মের কোনও হেরফের হয় না।

নিচে ON—GOTO ব্যবহার করে একটি প্রোগ্রামের উদাহরণ দেওয়া হল :

38

```
10  REM ·· WHO INVENTED TELEPHONE ?
20  PRINT "WHO INVENTED TELEPHONE ?"
30  PRINT "1) SAMUEL MORSE ?"
40  PRINT "2) ALEXANDER G. BELL ?"
50  PRINT "3) THOMAS A. EDISON ?"
60  PRINT "PLEASE TYPE 1, 2 OR 3" ;
70  PRINT "TO INDICATE CORRECT ANSWER"
80  INPUT Z
90  ON Z GOTO 100, 140, 170
100 PRINT "NO, SAMUEL MORSE INVENTED" ;
110 PRINT "THE MORSE CODE"
120 PRINT "TRY AGAIN !"
130 GOTO 60
140 PRINT "YES, ALEXANDER G. BELL" ;
150 PRINT "INVENTED THE TELEPHONE"
160 GOTO 210
170 PRINT "NO, THOMAS A. EDISON" ;
180 PRINT "INVENTED THE GRAMOPHONE"
190 PRINT "TRY AGAIN !"
200 GOTO 60
210 END
```

এই প্রোগ্রামের সাহায্যে একটি খেলার মাধ্যমে বের করা হচ্ছে টেলিফোনের আবিষ্কারক কে। ইচ্ছে করলে আবিষ্কার ও আবিষ্কারকের সংখ্যা বাড়িয়ে খেলাটিকে আরও আকর্ষণীয় করে তোলা যেতে পারে।

DIM

DIM মূল শব্দটির পুরো নাম হল DIMENSION—যার অর্থ মাপ। কোনও নির্দিষ্ট সারি অথবা সাংখ্যিক চলরাশির যদি বিভিন্ন মান হওয়ার সম্ভাবনা থাকে এবং আমরা যদি তার সব মানগুলো কাজে লাগাতে চাই বা সেগুলো নিয়ে কোনও তালিকা তৈরি করতে চাই, তাহলে DIM নির্দেশ সুবিধেজনক। যেমন,

10 DIM X(15)

এই উদাহরণটির অর্থ, 10 নম্বর নির্দেশটি পেলে কম্পিউটার X নামক সাংখ্যিক চলরাশিটির মোট 16টি মান (0 থেকে 15 পর্যন্ত) স্মৃতিকোষে সঞ্চয় করে রাখার জন্য তৈরি হবে। সাধারণত X(15) লিখলে আমরা X-এর 15টি মান (1 থেকে 15 পর্যন্ত) সঞ্চয় করি। এই মানগুলো X(1), X(2), ..., X(15) এই 15টি 'নামে' সঞ্চিত হয়ে থাকে। লক্ষ্য করার বিষয়, এখানে আমরা X(0)-কে বর্জন করেছি। প্রোগ্রাম লেখার সুবিধের জন্য বেশিরভাগ প্রোগ্রামারই এই নিয়মটি অনুসরণ করে।

DIM নির্দেশ প্রোগ্রামের যেখানে খুশি দেওয়া যায়। তবে প্রোগ্রামের শুরুতে এই নির্দেশ ব্যবহারের অভ্যেস থাকা ভাল। এর ফলে স্মৃতিকোষে কতটা জায়গা ব্যবহার করা হচ্ছে তা চট করে সহজেই দেখে নেওয়া যায়।

ওপরের উদাহরণের X(15)-কে বলা হয় অ্যারে ভ্যারিয়েবল বা বিন্যাস চলরাশি। অবশ্য আরও সঠিকভাবে বলতে গেলে X(15) হচ্ছে ওয়ান ডাইমেনশনাল অ্যারে ভ্যারিয়েবল বা একমাত্রিক বিন্যাস চলরাশি। এই ধরনের চলরাশি ব্যবহার করে নিচে একটি ছোট্ট প্রোগ্রামের উদাহরণ দেওয়া হল ঃ

39
```
10 REM ** INTRODUCTION TO ARRAY VARIABLE
20 DIM X(5)
30 FOR I=1 TO 5
40 READ X(I)
50 DATA 10, 20, 30, 40, 50
60 PRINT X(I);
70 NEXT I
80 END
```

এই প্রোগ্রামটি RUN করালে উত্তর পাওয়া যাবে :

```
RUN
  10  20  30  40  50
```

এর অর্থ, 20 নম্বর নির্দেশটির জন্য কম্পিউটারের স্মৃতিকোষে X নামের চলরাশির জন্য যে-পাঁচটি নির্দিষ্ট জায়গা রাখা ছিল তাতে এখন সঞ্চিত হয়ে গেছে 10, 20, 30, 40, 50—এই পাঁচটি মান। সুতরাং DIM নির্দেশের সুবিধে হল এই যে একটি মাত্র চলরাশি X-এর নাম ব্যবহার করে আমরা পাঁচটি বিভিন্ন মান সঞ্চয় করতে পারছি এবং প্রয়োজনে তা 'ছাপিয়ে' নিতে পারছি। নিচে একটা সহজ ছবি দিয়ে ঘটনাটি ব্যাখ্যা করা হল :

20 DIM X(5) নির্দেশটির জন্য কম্পিউটারের নির্দিষ্ট করা পাঁচটি স্মৃতিকোষের ক্রমিক সংখ্যা	1	2	3	4	5
স্মৃতিকোষগুলোতে সঞ্চিত X নামক বিন্যাস চলরাশিটির 5টি বিভিন্ন মান	10	20	30	40	50

ধরা যাক, একটি ট্যুরিস্ট বাস সার্ভিসের বাস দিনে একবার করে কলকাতা থেকে দীঘা যায়। সাধারণত ট্যুরিস্ট কোম্পানির কর্মচারীরা একটি রিজার্ভেশন চার্ট বা সংরক্ষণ তালিকা ব্যবহার করে। নতুন কোনও ক্রেতা টিকিট কাটতে এলে তারা দেখে ক্রেতা কোন্ দিনের কটা টিকিট চাইছে এবং সেই দিনে কোনও সিট খালি আছে কিনা। মনে করি, ডিসেম্বর মাসে তাদের বাসে প্রতিদিনই 7টি করে সিট খালি আছে। সুতরাং রিজার্ভেশন চার্ট দেখে তবেই তারা কোনও ক্রেতাকে নতুন টিকিট দেবে।

কম্পিউটার প্রোগ্রাম করে এই কাজটা করতে চাইলে DIM নির্দেশ ব্যবহার করা বেশ সুবিধেজনক। যেমন, প্রথমেই আমরা কম্পিউটারের স্মৃতিকোষে সঞ্চয় করে রাখতে চাই যে ডিসেম্বর মাসে প্রতিদিনই 7টি করে সিট খালি আছে। এর জন্য নিচের ছোট্ট প্রোগ্রামটি করতে হবে :

```
10  DIM  D(31)
20  FOR  I=1 TO 31
30  LET  D(I)=7
40  NEXT I
```

এই প্রোগ্রামটি করলে ডিসেম্বর মাসের 1 থেকে 31 তারিখ পর্যন্ত যে 7টি

করে সিট প্রতিদিন খালি রয়েছে সেটা কম্পিউটারের স্মৃতিতে D(31) নামক বিন্যাস চলরাশি হিসেবে সঞ্চিত থাকবে।

এখন কলকাতা-দীঘা ভ্রমণের বাস রিজার্ভেশন সংক্রান্ত সমস্যার সমাধান করতে হলে নিচের প্রোগ্রামটি করতে হবে ঃ

40

```
10  REM ∘∘ CALCUTTA-DIGHA BUS SERVICE
20  DIM D(31)
30  FOR I=1 TO 31
40  LET D(I)=7
50  NEXT I
60  PRINT
70  PRINT "TYPE THE DAY IN DEC.";
80  INPUT I
90  PRINT " NO. OF TICKETS REQUESTED";
100 INPUT N
110 IF D(I)>=N THEN 150
120 PRINT "SORRY ! ONLY "; D(I); " SEATS
                                  AVAILABLE";
130 PRINT "FOR DEC. "; I; ",MAKE
                        ANOTHER REQUEST PLEASE."
140 GOTO 60
150 PRINT "RESERVATION CONFIRMED.";
160 PRINT "ISSUE "; N; " TICKETS FOR
                                    DEC. "; I
170 LET D(I)=D(I)-N
180 PRINT "STILL "; D(I); " EMPTY SEATS
                                  ON DEC. "; I
190 PRINT "NEXT REQUEST PLEASE."
200 GOTO 60
210 END
```

এই উদাহরণটিতে শুধু ডিসেম্বর মাসের কথা বলা হয়েছে। ইচ্ছে করলে এই প্রোগ্রামকে উন্নত করে বছরের বারো মাসেরই রিজার্ভেশনের হিসেব রাখা যায়। গোটা বছরের রিজার্ভেশন চার্ট শুরু করার আগে বাসের সব

সিটই খালি আছে বলে ধরে নিতে হবে। প্রোগ্রামটিকে এইভাবে উন্নত করে আরও ক্ষমতাশালী করে তোলার দায়িত্ব পাঠকের ওপরে দেওয়া হল।

DIM নির্দেশের শেষ উদাহরণে আমরা 35 নম্বর প্রোগ্রামটিকেই উন্নত রূপ দেব। আর সেই সঙ্গে নতুন কয়েকটা মূল শব্দের অর্থও জেনে নেব।

35 নম্বর প্রোগ্রামে আমরা 250 জন ছাত্রের মধ্যে পরীক্ষায় সর্বোচ্চ নম্বর কত উঠেছে তা বের করেছি। কিন্তু কোন্ ছাত্র সর্বোচ্চ নম্বর পেয়েছে তা বের করা হয়নি। আমাদের নতুন প্রোগ্রাম সর্বোচ্চ নম্বর কত তা তো বের করবেই, উপরন্তু জানিয়ে দেবে কোন্ রোল নম্বরের ছাত্র সেই সর্বোচ্চ নম্বর পেয়েছে।

41

```
10  PRINT "TOTAL NO. OF STUDENTS=";
20  INPUT T
30  CLEAR
40  DIM M(250)
50  LET H=0
60  FOR R=1 TO 250
70  PRINT "ROLL NO.=";
80  INPUT R
90  PRINT "MARKS=";
100 INPUT M(250)
110 LET K=1000 * M(250) + R
120 IF K<=H THEN 140
130 LET H=K
140 NEXT R
150 LET A$=STR$ (H)
160 LET C$=RIGHT$ (A$, 3)
170 LET X=LEN (A$)
180 LET Y=X-3
190 LET B$=LEFT$ (A$, Y)
200 LET A=VAL (B$)
210 LET B=VAL (C$)
220 PRINT "HIGHEST MARKS="; A;
230 PRINT " OBTAINED BY ROLL NO."; B.
240 END
```

এই প্রোগ্রামটি আমরা একটু বিস্তারিতভাবে আলোচনা করব।

প্রথমে ধরা যাক 30 নম্বর লাইনের CLEAR নির্দেশটি। CLEAR হচ্ছে বেসিক ভাষার একটি গুরুত্বপূর্ণ সিস্টেম কমান্ড। প্রোগ্রাম চালাতে হলে আমরা যে RUN শব্দটি ব্যবহার করেছি সেটিও বেসিকের আর একটি সিস্টেম কমান্ড। পরে সপ্তম অধ্যায়ে আমরা আরও কয়েকটি বেসিক সিস্টেম কমান্ডের সঙ্গে পরিচিত হব। বেসিকের CLEAR নির্দেশটির অর্থ হচ্ছে স্মৃতিকোষ পরিষ্কার করা— যাতে কোনও সাংখ্যিক বা সারি চলরাশির পুরোনো কোনও সঞ্চিত মান প্রোগ্রামকে ভুল তথ্য সরবরাহ না করে। কোনও DIM নির্দেশ ব্যবহার করার আগে CLEAR নির্দেশ ব্যবহার করার অভ্যেস থাকা ভাল। অবশ্য ওপরের উদাহরণটিতে 30 নম্বর নির্দেশটি না দিলেও বিশেষ কোনও ক্ষতি নেই।

এর পরেই আমরা চলে যাব 110 নম্বর নির্দেশে। এই নির্দেশে কোনও ছাত্রের পরীক্ষার নম্বরকে 1000 দিয়ে গুণ করে রোল নম্বরের সঙ্গে যোগ করা হয়েছে। এই কৌশলটি করা হয়েছে কোনও ছাত্রের নম্বর ও রোল নম্বরকে জুড়ে একটি বড় সংখ্যা পাওয়ার জন্য। এই বড় সংখ্যাটি এমন হবে যে তার ডানদিকের শেষ তিনটে অঙ্ক সব সময়ে রোল নম্বর জানিয়ে দেবে, আর বাকি অঙ্কগুলো জানাবে পরীক্ষার নম্বর। যেমন, 122 রোল নম্বরের ছাত্রটি যদি পরীক্ষায় 87 নম্বর পেয়ে থাকে, তাহলে 110 নম্বর নির্দেশে K-এর মান হবে 87122। আবার 9 রোল নম্বরের ছাত্র যদি ফেল করে 18 পায় তাহলে তার বেলায় K-এর মান হবে 18009। এই দুটি ক্ষেত্রেই আমরা দেখতে পাচ্ছি K-এর শেষ তিনটে অঙ্ক সবসময় কোনও ছাত্রের রোল নম্বর জানাচ্ছে, আর বাকি অঙ্কগুলো জানিয়ে দিচ্ছে তার পাওয়া নম্বর। এছাড়া যে সবচেয়ে বেশি নম্বর পাবে তার ক্ষেত্রে K-এর মানও হবে সবচেয়ে বেশি।

এই ভাবে 250 জন ছাত্রের মধ্যে K-এর সর্বোচ্চ মান বের করে তা সঞ্চিত রাখা হল H চলরাশির মান হিসেবে। এবারে আমাদের কাজ হল, K সংখ্যাটিকে সঠিকভাবে দুভাগে ভাগ করে পরীক্ষার নম্বর ও রোল নম্বরকে আলাদা করে নেওয়া। এই কাজটিই করা হয়েছে 150 থেকে 210 নম্বর নির্দেশের মধ্যে। এই সাতটি নির্দেশে আমরা বেসিক ভাষার পাঁচটি নতুন শব্দ ব্যবহার করেছি ঃ STR$, RIGHT$, LEN, LEFT$ ও VAL। এগুলোকে বলা হয় লাইব্রেরি ফাংশান বা সঞ্চিত অপেক্ষক। প্রয়োজনে যখন খুশি এই অপেক্ষকগুলো ব্যবহার করা যায়। STR, LEN ও VAL যথাক্রমে STRING, LENGTH ও VALUE-এর সংক্ষিপ্ত রূপ।

150 নম্বর নির্দেশের STR$ (H) অংশটি সাংখ্যিক চলরাশি H-এর মানকে সারি চলরাশিতে পরিণত করছে। যদি H=87122 হয়, তাহলে 150 নম্বর নির্দেশ পাওয়ার পর 87122 সংখ্যাটি সংখ্যার 'ছবি' হয়ে A$-এর মান

হিসেবে কম্পিউটারের স্মৃতিকোষে সঞ্চিত হয়ে যাবে। অর্থাৎ, A$="87122" নির্দেশের মতো।

160 নম্বর নির্দেশের RIGHT$ (A$, 3) অংশ A$-এর ডান দিক থেকে তিনটি অঙ্ক (অর্থাৎ, 122) 'কেটে' নেবে। সুতরাং এই নির্দেশ পেলে C$-এর মান হিসেবে স্মৃতিকোষে সঞ্চিত হবে 122। RIGHT$-এর পাশে বন্ধনীর মধ্যে লেখা রয়েছে কোন্ সারি চলরাশির (এক্ষেত্রে A$) ডান দিক থেকে ক'টা অঙ্ক (এক্ষেত্রে 3) 'কেটে' নিতে হবে। অতএব দেখা যাচ্ছে, 160 নম্বর নির্দেশ পেলে C$-এর মান হিসেবে ছাত্রটির রোল নম্বর 122 সঞ্চিত হয়ে যাচ্ছে।

পরবর্তী নির্দেশে LEN (A$)-এর অর্থ A$ নামক সারি চলরাশিটির দৈর্ঘ্য কত—অর্থাৎ, মোট ক'টা অক্ষর রয়েছে A$-এ। সেটাই তখন হবে X নামক সাংখ্যিক চলরাশির মান। এক্ষেত্রে X-এর মান হবে 6, কারণ 87122 সংখ্যাটির অদৃশ্য + চিহ্নের জন্য (অর্থাৎ, +87122) LEN (A$)-এর উত্তর 6 হবে। সুতরাং 180 নম্বর নির্দেশে Y-এর মান হবে (6−3)=3।

190 নম্বর নির্দেশ, B$=LEFT$ (A$, Y)-এর অর্থ A$-এর বাঁদিক থেকে Y সংখ্যক (এক্ষেত্রে 3) অক্ষর 'কেটে' নিয়ে কম্পিউটার B$-এর মান হিসেবে সঞ্চয় করে রাখবে। সুতরাং A$ যদি +87122 হয় তাহলে B$=+87 হবে।

200 ও 210 নম্বর নির্দেশের VAL ফাংশানের কাজ হল শুধুই সংখ্যা দিয়ে তৈরি কোনও সারি চলরাশিকে সাংখ্যিক চলরাশিতে রূপান্তরিত করা। অর্থাৎ, STR$-এর ঠিক উল্টো। সুতরাং 200 ও 210 নম্বর নির্দেশের সাহায্যে B$ ও C$-এর মানগুলো সংখ্যার 'ছবি' থেকে রূপান্তরিত হবে প্রকৃত সংখ্যায়, এবং সঞ্চিত হবে যথাক্রমে A ও B নামক সাংখ্যিক চলরাশির মান হিসেবে।

সবশেষে A ও B-এর মান 'ছাপানোর' জন্যই কম্পিউটারকে নির্দেশ দেওয়া হয়েছে, এবং তখন উত্তর হিসেবে আমরা পাব :

HIGHEST MARKS= 87 OBTAINED BY ROLL NO. 122

সারি চলরাশির 'জটিলতার' মধ্যে না গিয়েও 41 নম্বর প্রোগ্রামটিকে লেখা যায়। যেমন, 150 থেকে 210 নম্বর নির্দেশের বদলে আমরা লিখতে পারি নিচের লাইনগুলো :

```
150   LET X=H/1000
160   LET A=INT(X)
170   LET B=(X-A)*1000
```

এক্ষেত্রে 180 থেকে 210 পর্যন্ত লাইনগুলোর কোনও প্রয়োজন নেই। আগের উদাহরণ অনুযায়ী মনে করি, H=87122। তাহলে 150 নম্বর নির্দেশ-মাফিক X=87·122। 160 নম্বর নির্দেশের ফলে আমরা পাব A=87, কারণ INT (X)-এর অর্থ, 'X-এর শুধু মাত্র পূর্ণসংখ্যার অংশটুকু রাখ, দশমিক অংশটুকু বর্জন কর।' তবে এই যে পূর্ণসংখ্যা, তা সব সময়েই X-এর মানের চেয়ে কম হবে—X ধনাত্মক বা ঋণাত্মক যা-ই হোক না কেন। যেমন,

$$INT(-0{\cdot}4) = -1$$
$$INT(-2{\cdot}6) = -3$$
$$INT(7{\cdot}9) = 7$$

INTEGER শব্দটির সংক্ষিপ্ত রূপ INT। INT হল বেসিকের আর একটি লাইব্রেরি ফাংশান।

অতএব 170 নম্বর নির্দেশে B-এর মান হবে

$$B = (87{\cdot}122 - 87) * 1000 = 122$$

এইভাবে আগের চেয়ে সহজে আমরা A ও B-এর মান পেয়ে গেলাম।

ওপরের উদাহরণে একটি সমস্যাকে দুভাবে করে দেখানোর অর্থ হল কোনও সমস্যাকে যে একাধিক পদ্ধতিতে কম্পিউটার প্রোগ্রাম করে সমাধান করা যায় তা বোঝানো। ওপরের উদাহরণে যদি পরীক্ষার নম্বর ও রোল নম্বরের সঙ্গে ছাত্রের নাম জানতে চাওয়া হত, তাহলে সমস্যাটি আরও কঠিন হয়ে দাঁড়াত। অল্পসংখ্যক ছাত্রের জন্য পাঠককে এরকম সমস্যা সমাধানের একটি বেসিক প্রোগ্রাম লিখতে অনুরোধ করছি। শুধু এটুকু বলে রাখি, প্রোগ্রামটি তৈরি করতে সারি চলরাশি ব্যবহার করতে হবে, এবং 41 নম্বর প্রোগ্রামের প্রথম সংস্করণটি এ-বিষয়ে পথ দেখাতে পারে।

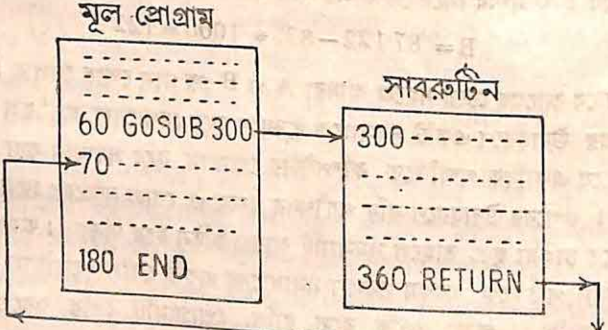
41 নম্বর প্রোগ্রামটির মাধ্যমে আমরা ছ'টি সঞ্চিত অপেক্ষকের সঙ্গে পরিচিত হলাম। পরের এই অধ্যায়ে রকমের আরও কয়েকটি অপেক্ষক নিয়ে আমরা আলোচনা করব।

GOSUB

GOSUB নির্দেশ নিয়ে আলোচনা করতে গেলে প্রথমেই আমাদের জানতে হবে সাবরুটিন কাকে বলে।

অনেক সময় কোনও একটি প্রোগ্রামে একই ধরনের গণনা একাধিক বার প্রয়োজন হয়। তখন সেই গণনার কাজটি যদি আমরা সাধারণ একটি বেসিক প্রোগ্রাম লিখে সম্পন্ন করতে চাই তাহলে প্রোগ্রামটি অনেক লম্বা হয়ে যাবে।

কারণ ওই বিশেষ গণনার কাজটির জন্য যে-ক'টি নির্দেশ প্রয়োজন তা হয়ত লিখতে হবে বার বার। প্রোগ্রামকে সংক্ষিপ্ত করার জন্য কোনও কোনও সময়ে FOR—NEXT লুপ ব্যবহার করা হয়। এই ধরনের কয়েকটি সমস্যা আমরা FOR—NEXT লুপ ব্যবহার করে সমাধান করেছি। কিন্তু সেক্ষেত্রে FOR—NEXT লুপের অংশটি মূল বেসিক প্রোগ্রামেরই অঙ্গ ছিল। যদি কোনও ভাবে FOR—NEXT লুপের অংশটিকে আমরা আলাদা একটি বেসিক প্রোগ্রাম হিসেবে লিখতে পারি, এবং মূল প্রোগ্রাম থেকে প্রয়োজন মতো ওই প্রোগ্রামে যেতে পারি এবং কাজ শেষ করে মূল প্রোগ্রামে আবার ফিরে আসতে পারি, তাহলে সেই প্রোগ্রামকে আমরা বলি সাবরুটিন (SUBROUTINE)। মূল প্রোগ্রাম থেকে সাবরুটিনে যাবার জন্য যে-নির্দেশ ব্যবহার করা হয় তাকেই বলে GOSUB। GOSUB হল GO TO SUBROUTINE-এর সংক্ষিপ্ত রূপ। নিচে মূল প্রোগ্রাম ও সাবরুটিনের গঠনগত সম্পর্ক দেখানো হল :

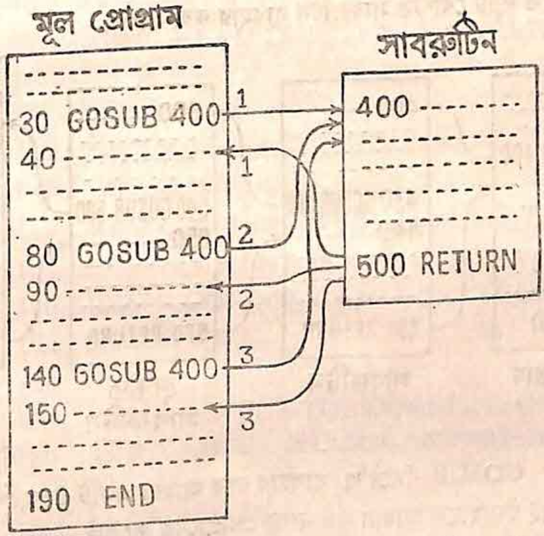


ওপরের উদাহরণে মূল প্রোগ্রামের 60 নম্বর নির্দেশে পেঁছে 'GOSUB 300' নির্দেশ পেয়ে কম্পিউটারের নিয়ন্ত্রণ সাবরুটিনের প্রথম লাইনে—300 নম্বর নির্দেশে—স্থানান্তরিত হচ্ছে। সেখানে সে পর পর সাবরুটিনের নির্দেশগুলো পালন করাচ্ছে কম্পিউটারকে দিয়ে। তারপর সাবরুটিনের শেষ লাইনে—360 নম্বর লাইনে—এসে নির্দেশ পাচ্ছে RETURN। RETURN নির্দেশের অর্থ, 'মূল প্রোগ্রামের GOSUB নির্দেশের ঠিক পরের নির্দেশটিতে ফিরে যাও এবং পর পর মূল প্রোগ্রামের বাকি নির্দেশগুলো পালন কর।' এখানে RETURN নির্দেশ পেয়ে কম্পিউটার নিয়ন্ত্রণ চলে আসবে মূল প্রোগ্রামের 70 নম্বর নির্দেশে এবং কম্পিউটার পর পর বাকি নির্দেশগুলো পালন করে যাবে।

যে কোনও সাবরুটিনের শেষ নির্দেশ সব সময়েই RETURN নির্দেশ হওয়া চাই। তা না হলে কম্পিউটার-নিয়ন্ত্রণ ঠিক ভাবে মূল প্রোগ্রামে ফিরে

আসতে পারবে না। তবে একটি সাবরুটিনে একাধিক RETURN নির্দেশ থাকতে পারে—যার অর্থ, বিভিন্ন শর্ত সাপেক্ষে কম্পিউটারের নিয়ন্ত্রণ সাবরুটিনের বিভিন্ন নির্দেশ থেকে মূল প্রোগ্রামে ফিরে আসতে পারে। কোনও সাবরুটিনের নির্দেশ পালন করার সময় কম্পিউটার-নিয়ন্ত্রণ যেই প্রথম RETURN নির্দেশটি পাবে তক্ষুনি সে ফিরে আসবে মূল প্রোগ্রামে।

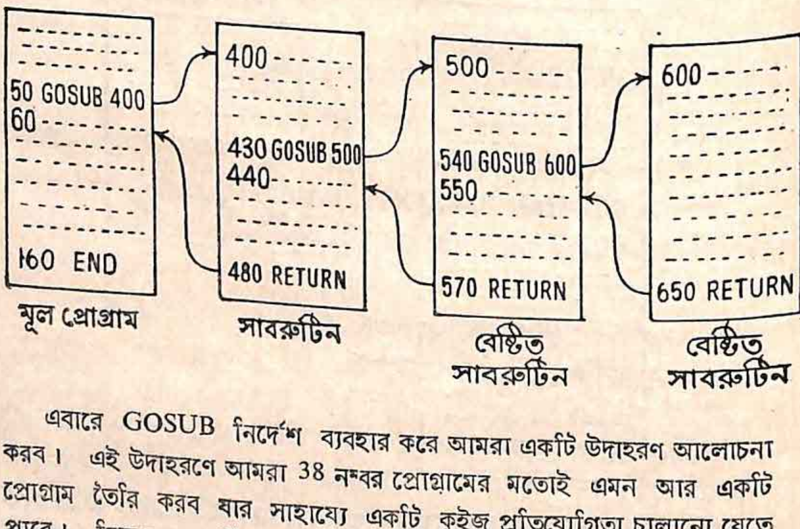
মূল প্রোগ্রামে GOSUB নির্দেশ দেবার সময়ে লক্ষ্য রাখতে হবে GOSUB মূল শব্দের পরে যেন অবশ্যই সাবরুটিনের প্রথম লাইন নম্বরটি লেখা হয়। যেমন, আগের পৃষ্ঠার উদাহরণটির ক্ষেত্রে 'GOSUB 300' লেখা হয়েছে। এছাড়া GOSUB নির্দেশের সাহায্যে কোনও মূল প্রোগ্রামের বিভিন্ন লাইন থেকে একই সাবরুটিনে বার বার যাওয়া যায় এবং মূল প্রোগ্রামে ফিরে আসা যায়। যে-GOSUB নির্দেশটি নিয়ন্ত্রণকে সাবরুটিনে পাঠাবে, সাবরুটিনের RETURN নির্দেশ সব সময় ঠিক তার পরের লাইনে নিয়ন্ত্রণকে ফেরৎ পাঠিয়ে দেবে। নিচে এই ধরনের একটি প্রোগ্রামের গঠন দেখানো হল ঃ



ওপরের উদাহরণে মূল প্রোগ্রাম থেকে একটি সাবরুটিনকে তিন বার ডাকা হয়েছে—যথাক্রমে 30, 80 ও 140 নম্বর নির্দেশ থেকে। আর প্রতিবারেই সাবরুটিনের RETURN নির্দেশ নিয়ন্ত্রণকে মূল প্রোগ্রামের যথাক্রমে 40, 90 ও 150 নম্বর নির্দেশে ফেরৎ পাঠিয়েছে।

এর আগে আমরা নেস্টেড লুপ বা বেষ্টিত লুপ নিয়ে আলোচনা করেছি। সাবরুটিনের ক্ষেত্রেও একাধিক বেষ্টিত সাবরুটিন বা নেস্টেড সাবরুটিন ব্যবহার

করা যায়। অর্থাৎ, ধরা যাক, একটি মূল প্রোগ্রাম থেকে GOSUB নির্দেশ ব্যবহার করে একটি সাবরুটিনে যাওয়া হল। ইচ্ছে করলে সেই সাবরুটিন থেকে আবার GOSUB নির্দেশ ব্যবহার করে দ্বিতীয় একটি সাবরুটিনে যাওয়া যাবে। এই দ্বিতীয় সাবরুটিনকে বলা হয় বেষ্টিত সাবরুটিন। দ্বিতীয় সাবরুটিনটি থেকে GOSUB ব্যবহার করে আবার আমরা চলে যেতে পারি তৃতীয় সাবরুটিনে। এটিও একটি বেষ্টিত সাবরুটিন। অর্থাৎ, প্রথম সাবরুটিন ছাড়া বাকি সবই হবে বেষ্টিত সাবরুটিন। সাধারণত বেসিক ভাষায় এগারোটি কিংবা তারও বেশি বেষ্টিত সাবরুটিন ব্যবহার করা যায়। এক্ষেত্রে একটা জরুরী কথা মনে রাখা দরকার ঃ কম্পিউটার-নিয়ন্ত্রণ যে-মূল প্রোগ্রাম থেকে কোনও সাবরুটিনে যাবে তাকে সব সময় সেই মূল প্রোগ্রামেই ফিরে আসতে হবে। আবার কোনও সাবরুটিন বা বেষ্টিত সাবরুটিন থেকে অন্য কোনও বেষ্টিত সাবরুটিনে গেলেও একই নিয়ম মানতে হবে। অর্থাৎ, যে-প্রোগ্রাম থেকে GOSUB নির্দেশ দিয়ে নিয়ন্ত্রণকে বেষ্টিত সাবরুটিনে পাঠানো হবে, তাকে আবার সেই প্রোগ্রামেই ফিরে আসতে হবে। নিচে এই নিয়মটি ছবির সাহায্যে দেখানো হল। ছবিতে একটি সাবরুটিন ও দুটি বেষ্টিত সাবরুটিন ব্যবহার করা হয়েছে।



এবারে GOSUB নির্দেশ ব্যবহার করে আমরা একটি উদাহরণ আলোচনা করব। এই উদাহরণে আমরা 38 নম্বর প্রোগ্রামের মতোই এমন আর একটি প্রোগ্রাম তৈরি করব যার সাহায্যে একটি কুইজ প্রতিযোগিতা চালানো যেতে পারে। বিজ্ঞানের ছ'টি আবিষ্কার ও তাদের আবিষ্কারকের নাম দিয়ে প্রোগ্রামটি দেখানো হল। ইচ্ছে করলে পাঠক প্রশ্নের সংখ্যা বাড়াতে পারে।

উদাহরণের ছ'টি প্রশ্ন হল ঃ

১। টেলিফোন কে আবিষ্কার করেন ?

২। মর্স কোড কে আবিষ্কার করেন ?

৩। গ্রামোফোন কে আবিষ্কার করেন ?

৪। বাষ্পচালিত ইঞ্জিন কে আবিষ্কার করেন ?

৫। কালাজ্বরের ওষুধ কে আবিষ্কার করেন ?

৬। এ. সি. মোটর কে আবিষ্কার করেন ?

নিচে এই কুইজ প্রতিযোগিতার প্রোগ্রামটি দেওয়া হল ঃ

42

```
10  REM •• QUIZ COMPETITION PROGRAM
20  CLEAR
30  DIM Q$(6)*20, A$(6)*20, X$(6)*20
40  REM •• SET SCORE TO ZERO
50  LET S=0
60  REM •• SET  UP LOOP  FOR COUNTING NO.
                               OF QUESTIONS
70  FOR N=1 TO 6
80  REM •• READ QUESTION & CORRECT
                                     ANSWER
90  READ Q$ (N), A$(N)
100 REM •• ASK QUESTION
110 GOSUB 500
120 REM •• WAS ANSWER CORRECT ?
130 IF F=0 THEN 190
140 REM •• ASK QUESTION FOR SECOND TIME
150 GOSUB 500
160 IF F=0 THEN 190
170 REM •• OUTPUT THE CORRECT ANSWER
180 PRINT "THE CORRECT ANSWER IS " ; A$(N)
190 NEXT N
200 PRINT "YOUR SCORE IS " ; S ; " OUT OF 6"
210 STOP
220 DATA "TELEPHONE", "ALEXANDER G.
                                      BELL"
230 DATA "MORSE CODE", "SAMUEL MORSE"
240 DATA "GRAMOPHONE", "THOMAS
                                 A. EDISON"
```

```
250  DATA "STEAM ENGINE", "JAMES WATT"
260  DATA "KALA-AZAR", "U. BRAHMACHARI"
270  DATA "AC MOTOR", "NICOLA TESLA"

499  REM ∞ SUBROUTINE TO ASK QUESTION
500  LET F=O
510  REM ∞ OUTPUT QUESTION
520  PRINT "WHO INVENTED THE "; Q$(N)
530  INPUT X$(N)
540  IF A$(N)=X$(N) THEN 580
550  PRINT "SORRY ! YOUR ANSWER IS WRONG."
560  LET F=1
570  GOTO 600
580  PRINT "YOUR ANSWER IS CORRECT."
590  LET S=S+1
600  RETURN
```

ওপরের প্রোগ্রামে লাইন নম্বর 10 থেকে 270 পর্যন্ত মূল প্রোগ্রাম, আর লাইন নম্বর 500 থেকে 600 পর্যন্ত হল সাবরুটিন। 499 নম্বর লাইনটি একটি REM নির্দেশ। এটি ব্যবহার করা হয়েছে প্রোগ্রামারের সুবিধের জন্য। এই লাইনটি সাবরুটিনের অংশ হলেও কোনও ক্ষতি ছিল না। তখন 110 ও 150 নম্বর লাইনের GOSUB নির্দেশে 500 সংখ্যাটির বদলে 499 লিখতে হত।

এই উদাহরণটিতে একাধিক REM নির্দেশ ব্যবহার করা হয়েছে প্রোগ্রাম বোঝার সুবিধের জন্য।

প্রোগ্রামটির আলোচনা শুরু করা যাক 30 নম্বর নির্দেশটি দিয়ে। আগে সাংখ্যিক চলরাশির ক্ষেত্রে DIM নির্দেশের ব্যবহার আমরা দেখেছি। কিন্তু এবারে যেহেতু আমাদের কুইজের প্রশ্ন ও উত্তর কোনও সংখ্যা নয়, নাম, সেহেতু আমরা সারি চলরাশির উপযুক্ত DIM নির্দেশ ব্যবহার করেছি। DIM Q$(6)*20 নির্দেশটির অর্থ হল, Q$ নামক সারি চলরাশির জন্য 6টি মান স্মৃতিকোষে সঞ্চয়ের ব্যবস্থা রাখতে হবে। কিন্তু Q$-এর এক-একটি মান দৈর্ঘ্যে কত বড় তা না জানালে কম্পিউটার জায়গা রাখবে কেমন করে? সেইজন্যেই Q$(6)-এর পরে * চিহ্ন দিয়ে 20 সংখ্যাটি জানানো হয়েছে। অর্থাৎ, Q$-এর এক-একটি মান কখনওই 20টি অক্ষরের বেশি হবে না। Q$-

এর মধ্যে কোনও শূন্যস্থান ব্যবহার করলে তাকেও দৈর্ঘ্যের হিসেবের মধ্যে ধরতে হবে। ঠিক একই ভাবে 30 নম্বর নির্দেশে আমরা A$ ও X$-এর জন্য জায়গা রেখেছি।

প্রোগ্রামটিতে মোট 6টি প্রশ্ন করা হয়েছে (70 নম্বর নির্দেশ) এবং কম্পিউটারকে প্রথমেই সঠিক প্রশ্ন (Q$) ও উত্তর (A$) পড়ে নিতে নির্দেশ দেওয়া হয়েছে (90 নম্বর নির্দেশ)। তার পরেই নিয়ন্ত্রণ চলে গেছে সাবরুটিনে প্রথম প্রশ্ন করতে (520 নম্বর নির্দেশ)। তখন ভি. ডি. ইউ.-তে যে-জিজ্ঞাসা-চিহ্ন ফুটে উঠছে তার উত্তরে কুইজ প্রতিযোগীরা কী-বোর্ড থেকে বোতাম টিপে তাদের উত্তর (X$) কম্পিউটারকে জানাচ্ছে (530 নম্বর নির্দেশ)। এরপর 540 নম্বর নির্দেশ প্রতিযোগীর জানানো উত্তর (X$) সঠিক উত্তরের (A$) সঙ্গে তুলনা করছে এবং প্রতিযোগীর উত্তর ঠিক কি ভুল তা ভি. ডি. ইউ. মারফৎ প্রতিযোগীকে জানিয়ে দিচ্ছে। উত্তর ঠিক হলে প্রতিযোগী 1 পয়েন্ট পাচ্ছে, আর ভুল হলে কোনও পয়েন্ট নেই। এই পয়েন্ট দেবার ব্যাপারটি করছে চলরাশি S। প্রোগ্রামের শুরুতে S-এর মান 0 রাখা হয়েছে (50 নম্বর নির্দেশ), কিন্তু উত্তর ঠিক হলে S এর মান প্রতিটি সঠিক উত্তরের জন্য 1 করে বেড়ে যাবে (590 নম্বর নির্দেশ)। কোনও প্রশ্নের উত্তর ভুল হলে প্রতিযোগীকে দ্বিতীয়বার উত্তর দেবার সুযোগ দেওয়া হচ্ছে। উত্তর ঠিক না ভুল তা কম্পিউটার বুঝতে পারছে F নামক চলরাশির মান থেকে (500 ও 560 নম্বর নির্দেশ)। F-এর মান 0 হলে উত্তর সঠিক, আর F = 1 হলে উত্তর ভুল। 130 ও 160 নম্বর নির্দেশে কম্পিউটার F-এর মান খতিয়ে দেখে তবে সিদ্ধান্ত নিচ্ছে।

প্রোগ্রামের শেষে কম্পিউটার কোনও প্রতিযোগীর মোট পয়েন্ট, অর্থাৎ S-চলরাশির সর্বোচ্চ মান, ভি. ডি. ইউ.-র মাধ্যমে প্রতিযোগীকে জানিয়ে দিচ্ছে (200 নম্বর নির্দেশ)।

প্রোগ্রামের 220 থেকে 270 নম্বর নির্দেশ হল DATA নির্দেশ। এই নির্দেশগুলোর মাধ্যমেই পর পর Q$(N) ও A$(N)-এর মান জানানো হয়েছে। কী-বোর্ড থেকে উত্তর দেবার সময়ে প্রতিযোগীদের একটা বিষয়ে লক্ষ্য রাখতে হবে। DATA নির্দেশগুলোতে A$(6) যেভাবে লেখা আছে হুবহু সেইভাবেই আবিষ্কারকের নামগুলো লিখতে হবে। কারণ 540 নম্বর নির্দেশে কম্পিউটার যখন A$(N) ও X$(N)-এর মান তুলনা করবে, তখন সে 30 নম্বর নির্দেশ মতো দুটি সারি চলরাশির 20টি করে অক্ষর প্রথম থেকে প্রতিটি অক্ষর ধরে ধরে পরস্পর তুলনা করবে। এই তুলনায় সামান্য গরমিল হলেই সে

জানিয়ে দেবে উত্তর ভুল। নিচে তুলনার পদ্ধতিটি ছবির সাহায্যে দেখানো হল :

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A$ (1) =	A	L	E	X	A	N	D	E	R		G	.		B	E	L	L			

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
X$ (1) =	A	L	E	X	A	N	D	E	R		G	.		B	E	L	L			

ওপরের ছবিতে A$ ও X$-এর 20টি অক্ষরের জন্য স্মৃতিকোষের কাল্পনিক 20টি ঘর দেখানো হয়েছে। 42 নম্বর প্রোগ্রামের 540 নম্বর নির্দেশে A$(N) ও X$(N)-এর তুলনার অর্থ ছবির 1 থেকে 20 পর্যন্ত প্রতিটি ঘর পরস্পরের সঙ্গে তুলনা করা। এই তুলনায় যদি প্রতিটি অক্ষর (যেমন 1 থেকে 9, 11 এবং 14 থেকে 17 নম্বর ঘর). চিহ্ন (যেমন, 12 নম্বর ঘরের ফুলস্টপ) এবং শূন্যস্থান (যেমন 10, 13 ও 18 থেকে 20 নম্বর ঘর) পরস্পরের সঙ্গে হুবহু মেলে তবেই কম্পিউটার মেনে নেবে A$(N) = X$(N), নচেৎ নয়।

GOSUB ব্যবহার করে এরকমই আরও সুন্দর সুন্দর প্রোগ্রাম তৈরি করা যায়। এই ধরনের কয়েকটি প্রোগ্রাম তৈরি করে দেখতে পাঠককে অনুরোধ করছি। ইচ্ছে করলে সেখানে বেষ্টিত সাবরুটিনও ব্যবহার করা যেতে পারে।

ON—GOSUB

ON—GOSUB নির্দেশের ব্যবহার হুবহু ON—GOTO-এর মতোই। শুধু ON—GOSUB-এর ক্ষেত্রে নিয়ন্ত্রণ মূল প্রোগ্রামের অন্য কোনও লাইনে না গিয়ে বিভিন্ন সাবরুটিনের প্রথম লাইনে স্থানান্তরিত হবে। যেমন,

```
130  ON K GOSUB 300, 500, 400, 700
140  ......................
```

এখানে K-এর মান 1 হলে নিয়ন্ত্রণ স্থানান্তরিত হবে 300 নম্বর লাইন থেকে যে-সাবরুটিন শুরু হচ্ছে সেখানে। আবার K = 2 হলে 500 নম্বর লাইন থেকে যে-সাবরুটিনের শুরু নিয়ন্ত্রণ চলে যাবে সেখানে। তেমনি K যদি 3 কিংবা 4 হয়, তাহলে নিয়ন্ত্রণ তার গন্তব্য হিসেবে বেছে নেবে যথাক্রমে 400 ও 700 নম্বর লাইন থেকে শুরু হওয়া সাবরুটিন দুটি। সুতরাং ওপরের উদাহরণের 140 নম্বর নির্দেশটি পালিত হবে নিয়ন্ত্রণ সাব-

রুটিনের কাজ সেরে ফিরে আসার পর—তা সে চারটি সাবরুটিনের যেটাই বেছে নিক না কেন।

ON—GOTO-এর সঙ্গে তুলনা করে দেখলে ON—GOSUB-এর ব্যবহার পাঠকের কাছে আরও পরিষ্কার হয়ে যাবে।

বেসিক ভাষার প্রয়োজনীয় এবং গুরুত্বপূর্ণ মূল শব্দগুলোর আলোচনা আমরা এখানেই শেষ করছি। বেসিকে অন্যান্য আরও যেসব মূল শব্দ আছে তার জন্য পাঠককে কম্পিউটার ম্যানুয়াল এবং বেসিক ভাষার অন্যান্য বই দেখতে অনুরোধ করছি।

পরবর্তী অধ্যায়ে আমরা বেসিক ভাষার লাইব্রেরি ফাংশান বা সঞ্চিত অপেক্ষক নিয়ে আলোচনা করব।

৬ □ বেসিক ঃ লাইব্রেরি ফাংশান

এর আগের অধ্যায়ে আমরা বেসিকের ছ'টি সঞ্চিত অপেক্ষক বা লাইব্রেরি ফাংশান নিয়ে সংক্ষেপে আলোচনা করেছি। এখন আরও কয়েকটি দরকারী লাইব্রেরি ফাংশানের সঙ্গে আমরা পরিচিত হব।

ABS

ABSOLUTE শব্দটির সংক্ষিপ্ত রূপ হচ্ছে ABS। ABS(X)-এর অর্থ হচ্ছে, X চলরাশিটির যথার্থ মান। এই মান সর্বদাই ধনাত্মক। যেমন,

| 43 |

```
10  LET  X = -5·3
20  LET  Y = ABS(X)
30  PRINT Y
40  END
```

এই প্রোগ্রামটির উত্তর হল 5·3। কারণ, ABS(X) = ABS(−5·3) = 5·3। সুতরাং X-এর মান ঋণাত্মক বা ধনাত্মক যাই হোক না কেন, ABS(X) সর্বদাই ধনাত্মক।

SQR

SQUARE ROOT কথাটির সংক্ষিপ্ত রূপ হল SQR। সুতরাং SQR-এর অর্থ হল বর্গমূল নির্ণয়। যেমন, SQR(25) = $\sqrt{25}$ = 5। যদি আমরা লিখি,

```
10  LET  A = SQR(B)
```

তাহলে এর অর্থ হল B চলরাশিটির বর্গমূল A চলরাশির মান হিসেবে কম্পিউটারের স্মৃতিকোষে সঞ্চিত থাকবে। সন্দেহ নেই ওপরের নির্দেশটিকে আমরা এইভাবেও লিখতে পারতাম ঃ

```
10  LET A = B ↑ 0·5
```

সুতরাং সেদিক থেকে চিন্তা করলে SQR লাইব্রেরি ফাংশানটির সত্যি সত্যি কোনও দরকার নেই। এটাকে একটা বাড়তি সুবিধে হিসেবে ভাবা যেতে পারে। তাছাড়া ঘাত সংকারকের চেয়ে SQR ফাংশান ব্যবহার করে বর্গমূল নির্ণয় করলে কোনও কোনও ক্ষেত্রে কম্পিউটারের হয়ত সময় কম লাগতে পারে।

কিন্তু যদি কখনও আমরা লিখি SQR(−25) তবে তা হবে অর্থহীন। এই ধরনের নির্দেশ কম্পিউটারকে দিলে কম্পিউটার জানিয়ে দেবে, 'ভুল হয়েছে।' তবে অনেক কম্পিউটারে 'ভুল হয়েছে' খবরটা জানানোর সঙ্গে সঙ্গে নির্দেশটিকে SQR(25) ধরে নিয়ে উত্তর হিসেবে 5 সংখ্যাটি জানিয়ে দেবার ব্যবস্থা থাকে। কিন্তু যদি লেখা হয়,

SQR (ABS (−25))

তাহলে গোলমালের কোনও সম্ভাবনা নেই। তখন সহজেই নির্ভুল উত্তর পাওয়া যাবে 5।

SGN

ধনাত্মক বা ঋণাত্মক চিহ্ন, অর্থাৎ, SIGN নির্ণয়ের জন্য SGN (পুরো নাম SIGNUM) ফাংশানটি ব্যবহার করা হয়। যেমন,

```
10  LET  A=SGN (B)
```

এই নির্দেশটির অর্থ হল, B যদি কোনও ধনাত্মক সংখ্যা হয় তাহলে A চলরাশির মান হবে +1, B যদি 0 হয় তাহলে A-এর মানও হবে 0, আর B যদি ঋণাত্মক সংখ্যা হয় তাহলে A=−1। অর্থাৎ, B-এর সাংখ্যিক মান যাই হোক না কেন, তা নিয়ে A-এর মানের কোনও হেরফের হবে না। শুধু B-এর মানের ধনাত্মক বা ঋণাত্মক চিহ্নের ওপর নির্ভর করেই A-এর মান +1 বা −1 হবে। আর B-এর মান 0 হলে A-এর মানও হবে 0।

RND

অনেক আকর্ষণীয় কম্পিউটার প্রোগ্রামে—যেমন, লটারি বা ওই জাতীয় কোনও খেলায়—র‍্যাণ্ডম নাম্বার বা এলোমেলো সংখ্যার প্রয়োজন হয়। RANDOM শব্দটির সংক্ষিপ্ত রূপ হল RND। এই RND ফাংশানের সাহায্যে কম্পিউটার 0 থেকে 1 এর মধ্যে একটি অনিয়মিত বা এলোমেলো সংখ্যা ভি ডি. ইউ.-তে ফুটিয়ে তোলে। যতবার RND ফাংশান ব্যবহার করা হবে ততবারই এক একটি নতুন এলোমেলো সংখ্যা ফুটে উঠবে পর্দায়। যেমন, যদি লেখা হয়,

44

```
10  LET A=RND
20  PRINT A
30  END
```

তাহলে উত্তর পাওয়া যাবে একটি এলোমেলো সংখ্যা। এই প্রোগ্রামটি

যতবার RUN করানো হবে ততবারই নতুন নতুন সংখ্যা পাওয়া যাবে। এই রকম তিনটি উত্তর নিচে দেওয়া হল ঃ

0·1011204657, 0·2504688473, 0·9589419601

MID$

এর আগে আমরা RIGHT$ ও LEFT$-এর সঙ্গে পরিচিত হয়েছি। MID$ হচ্ছে এই দুটি ফাংশানেরই এক ধরনের সম্মিলিত রূপ। MID$ (A$, 4, 3)-এর অর্থ, A$ নামক সারি চলরাশির 4 নম্বর অক্ষর থেকে পর পর 3টি অক্ষর গ্রহণ করা। নিচে MID$ ব্যবহারের একটি উদাহরণ দেওয়া হল ঃ

45

```
10 REM ** ILLUSTRATION OF MID$
20 LET A$="ADDRESS"
30 LET B$=MID$ (A$, 3, 5)
40 PRINT A$, B$
50 END
```

এই প্রোগ্রামটি RUN করালে উত্তর পাওয়া যাবে ঃ

```
RUN
ADDRESS
DRESS
```

এখানে 30 নম্বর নির্দেশের অর্থ A$ নামক সারি চলরাশির 3 নম্বর অক্ষর বা চিহ্ন থেকে পরপর 5টি অক্ষর বা চিহ্ন B$-এর মান হিসেবে সঞ্চিত হবে। সুতরাং ADDRESS শব্দটির তৃতীয় অক্ষর D থেকে পরপর পাঁচটি অক্ষর (DRESS) B$-এর মান হিসেবে 'ছাপা' হয়েছে।

EXP

EXP-এর পুরো কথাটি হল EXPONENTIAL। অর্থাৎ, EXP নামক লাইব্রেরি ফাংশানটির অর্থ এক্সপোনেনশিয়াল ফাংশন। মনে করি, e এমন একটি চলরাশি যাকে নিচের গাণিতিক শ্রেণীটির সাহায্যে প্রকাশ করা যায় ঃ

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots\cdots + \frac{x^n}{n!} + \cdots\cdots \text{(অসীম পর্যন্ত)}$$

এই শ্রেণীটিতে x-এর মান অনুযায়ী e^x-এর মান বাড়বে বা কমবে। এই শ্রেণীটিকে বলা হয় এক্সপোনেনশিয়াল সিরিজ বা এক্সপোনেনশিয়াল শ্রেণী। যদি $x=1$ হয় তাহলে ওপরের শ্রেণীটিতে x-এর মান 1 বসিয়ে দেখা যায় যে $e^x = 2{\cdot}718281828$ (প্রায়)।

বেসিক ভাষায় EXP ফাংশানটি এক্সপোনেনশিয়াল মান নির্ণয় করে। যেমন,

10 LET X=EXP (2·0)

নির্দেশটি দিলে X-এর মান হিসেবে $e^2 = 7·389056099$ সঞ্চিত হবে।

x এর মানের বৃদ্ধির সঙ্গে সঙ্গে e^x-এর মান অত্যন্ত দ্রুত হারে বেড়ে ওঠে। যেমন, নিচের প্রোগ্রামটি EXP ফাংশানের সাহায্যে e^x-এর কয়েকটি মান নির্ণয় করছে ঃ

46

```
10 REM ∞ EXP FUNCTION
20 PRINT "N=";
30 INPUT N
40 FOR X=1 TO N
50 PRINT "EXP"; X; "="; EXP(X)
60 NEXT X
70 END
```

যদি N-এর মান 3 দেওয়া হয়, তাহলে কম্পিউটার উত্তর দেবে ঃ

```
RUN
N=? 3
EXP (1)=2·718281828
EXP (2)=7·389056099
EXP (3)=20·08553692
```

LOG

LOGARITHM শব্দটির সংক্ষিপ্ত রূপ LOG। বেসিক ভাষার LOG ফাংশানটি ন্যাচারাল লগারিদ্‌ম্‌ বা স্বাভাবিক লগারিদ্‌ম্‌ নির্ণয় করে। যেমন, মনে করি,

$$\log x = y$$

তাহলে, $$x = e^y$$

এখানে x ও y দুটি সংখ্যা, আর e=2·718281828 (প্রায়)। এই ধরনের লগারিদ্‌ম্‌-এ e কে বলা হয় লগারিদ্‌ম্‌-এর নিধান বা বেস।

যদি আমরা এইরকম একটি প্রোগ্রাম লিখি ঃ

47

```
10 REM ∞ LOG FUNCTION
20 PRINT "X=";
```

```
30 INPUT X
40 LET Y=LOG (X)
50 PRINT "LOG (X)="; Y
60 GOTO 20
70 END
```

তাহলে X-এর বিভিন্ন মানের জন্য LOG (X)-এর যথাযথ মান আমরা পেয়ে যাব।

SIN, COS, TAN, COT

SIN, COS, TAN, ও COT—ত্রিকোণমিতীয় এই ফাংশানগুলোর প্রধান সম্পর্ক সমকোণী ত্রিভুজের সঙ্গে।

ছবির সমকোণী ত্রিভুজের $\angle ABC = \theta$ (মনে করি)।

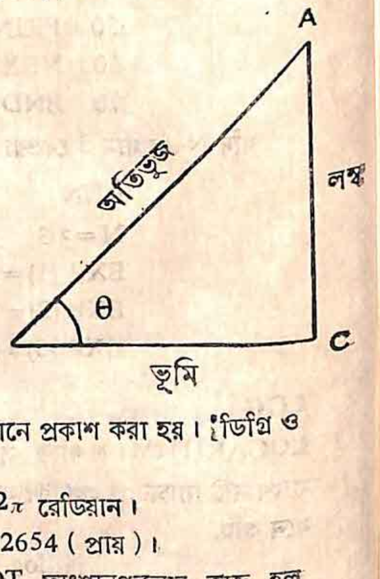
তাহলে সংজ্ঞা অনুসারে,

$$\text{SIN } \theta \text{ (বা } \sin\theta \text{)} = \frac{\text{লম্ব}}{\text{অতিভুজ}} = \frac{AC}{AB}$$

$$\text{COS } \theta \text{ (বা } \cos\theta \text{)} = \frac{\text{ভূমি}}{\text{অতিভুজ}} = \frac{BC}{AB}$$

$$\text{TAN } \theta \text{ (বা } \tan\theta\text{)} = \frac{\text{লম্ব}}{\text{ভূমি}} = \frac{AC}{BC}$$

$$\text{এবং COT } \theta \text{ (বা } \cot\theta\text{)} = \frac{\text{ভূমি}}{\text{লম্ব}} = \frac{BC}{AC}$$



এই θ কোণকে সাধারণত ডিগ্রি বা রেডিয়ানে প্রকাশ করা হয়। ডিগ্রি ও রেডিয়ানের মধ্যে সম্পর্ক হল :

360° (ডিগ্রি) = 4 সমকোণ = 2π রেডিয়ান।

এখানে π রাশিটির মান হল, $\pi = 3 \cdot 141592654$ (প্রায়)।

বেসিক ভাষায় SIN, COS, TAN ও COT ফাংশানগুলোর কাজ হল ওপরের সংজ্ঞা অনুযায়ী কোনও কোণের ক্ষেত্রে ত্রিকোণমিতীয় ফাংশানগুলোর (sin, cos, tan ও cot) যথাযথ মান নির্ণয় করা। যেমন,

```
10 LET Y=SIN (X)
20 LET Y=COS (X)
30 LET Y=TAN (X)
40 LET Y=COT (X)
```

এখানে X কোণটির মান সাধারণত রেডিয়ানে দেওয়া হয়। তবে বেসিকের কোনও কোনও সংস্করণে X-এর মান ডিগ্রি বা রেডিয়ান দু-রকম এককেই দেওয়া চলে। এখানে একটা কথা উল্লেখ করা প্রয়োজন ঃ বেসিকের সব সংস্করণে COT ফাংশানটি পাওয়া যায় না।

নিচের প্রোগ্রামটি X-এর দশটি বিভিন্ন মানের জন্য SIN (X)-এর যথাযথ মান নির্ণয় করছে ঃ

48

```
10 REM ** SIN FUNCTION
20 FOR I=1 TO 10
30 PRINT "X=";
40 INPUT X
50 LET Y=SIN (X)
60 PRINT "SIN (X)="; Y
70 NEXT I
80 END
```

ATN, ASN, ACS

যদি, TAN (X) = Y হয়, তাহলে সংজ্ঞা অনুযায়ী ARC TAN (Y) = X

এই ARC TAN (Y)-কেই বেসিক ভাষায় সংক্ষেপে ATN (Y) লেখা হয়। অর্থাৎ, ATN (Y) = X। সুতরাং, TAN (X)-এর মান জানা থাকলে ATN ফাংশান ব্যবহার করে X-কোণের মান নির্ণয় করা যায়। যেমন, আমরা জানি,

$$\text{TAN } (\pi/4) = 1{\cdot}0$$

সুতরাং, ATN $(1{\cdot}0) = \pi/4 = 0.7853981634$ (রেডিয়ান)

বেসিকের কোনও কোনও সংস্করণে কোণের মান ডিগ্রিতেও প্রকাশ করার ব্যবস্থা আছে।

ATN ফাংশানের মতো আরও দুটি ফাংশান ASN (ARC SIN) ও ACS (ARC COS) বেসিকের উন্নত সংস্করণে অনেক সময় পাওয়া যায়। এই ফাংশান দুটির ব্যবহার ATN-এরই মতো। অর্থাৎ,

SIN (X) = Y হলে ASN (Y) = X

এবং COS (X) = Y হলে ACS (Y) = X

TAB

PRINT নির্দেশের সাহায্যে আমরা ভি. ডি. ইউ.-তে লেখার কাজ করি। আর LPRINT নির্দেশ ব্যবহার করে লেখা হয় প্রিন্টারে। লেখার সময়ে কমা

ও সেমিকোলন ব্যবহার করে আমরা লেখা সাজানোর কাজ করেছি ঃ যেমন, দুটি উত্তর একই লাইনে কিছুটা ফাঁক দিয়ে লেখা, অথবা দুটি উত্তর পর পর গায়ে গায়ে লেখা, কিংবা দুটিকে দু-লাইনে লেখা ইত্যাদি। কিন্তু ভি. ডি. ইউ.-তে বা প্রিণ্টারে লেখা উত্তর কিংবা মন্তব্য খুশি মতো সাজানোর কাজে অতুলনীয় ক্ষমতাশালী একটি ফাংশান হল TAB ফাংশান। এই ফাংশান ব্যবহার করে কম্পিউটারের সাহায্যে লেখচিত্র, রেখাচিত্র অথবা নানান নকশা খুব সহজেই তৈরি করা যায়।

কোনও ভি. ডি. ইউ.-তে সাধারণত 80 টি অক্ষর বা চিহ্ন একটি লাইনে পাশাপাশি বসানো যায়। এই আশিটি অবস্থানকে TAB ফাংশান 0, 1, 2... 78, 79 পর্যন্ত আশিটি সংখ্যা দিয়ে নির্দিষ্ট করে দিতে পারে। TAB ফাংশান ব্যবহার করে কোনও PRINT নির্দেশ দিলে TAB ফাংশানের নির্দিষ্ট করে দেওয়া অবস্থান থেকেই কম্পিউটার ভি. ডি. ইউ.-তে লেখার কাজ শুরু করবে। যেমন,

10 PRINT TAB (10) ; "USE OF TAB FUNCTION"

এই নির্দেশ দিলে ভি. ডি. ইউ.-তে প্রথম 10টি ঘর (0 থেকে 9 নম্বর অবস্থান পর্যন্ত) ছেড়ে 11 নম্বর ঘর থেকে (অর্থাৎ, 10 নম্বর অবস্থান থেকে) USE OF TAB FUNCTION লেখাটি ফুটে উঠবে। যদি কোনও কারণে নির্দেশটি এরকম হয়,

10 PRINT "USE OF TAB" ; TAB (8) ; "FUNCTION"

তাহলে সামান্য অসুবিধে দেখা দেবে। কারণ, প্রথমে কম্পিউটার ভি. ডি. ইউ.-তে প্রথম ঘর (অর্থাৎ, 0 নম্বর অবস্থান) থেকেই USE OF TAB শব্দ তিনটে লিখে ফেলবে। লক্ষ্য করলেই দেখা যাবে যে এই শব্দ তিনটে লিখতে মোট 10টি ঘর লেগেছে (শব্দের মাঝের শূন্যস্থান 2টি ধরে)। সুতরাং, তারপরই TAB (8) নির্দেশ অনুযায়ী 8টি ঘর ছেড়ে 9 নম্বর ঘর থেকে FUNCTION শব্দটি লিখতে গিয়ে কম্পিউটার নির্দিষ্ট ঘরটি খালি পাবে না। কারণ সেখানে A অক্ষরটি লেখা রয়েছে, আর তার পরেই 10 নম্বর ঘরে লেখা রয়েছে B অক্ষরটি। অতএব তখন সে 9-এর বদলে 11 নম্বর ঘর থেকেই FUNCTION শব্দটি লিখে দেবে। ফলে উত্তরটা হবে এই রকম ঃ

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
U S E   O F   T A B    F  U  N  C  T  I  O  N
```

বেসিকের কোনও কোনও সংস্করণে এই ধরনের নির্দেশ দেওয়া হলে

কম্পিউটার পরের লাইন থেকে TAB ফাংশান অনুযায়ী লেখার কাজ শুরু করে। সেক্ষেত্রে ভি. ডি. ইউ.-তে ফুটে উঠবে ঃ

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
U S E   O F   T A B
                F U  N  C  T  I  O  N
```

যদি লেখা হয়,

10 PRINT TAB (4·32) ; "USE OF TAB FUNCTION"

তাহলে 4·32 সংখ্যাটির দশমিক অংশ কম্পিউটার ছেঁটে ফেলবে এবং নির্দেশটিকে ধরে নেবে ঃ

10 PRINT TAB (4) ; "USE OF TAB FUNCTION"

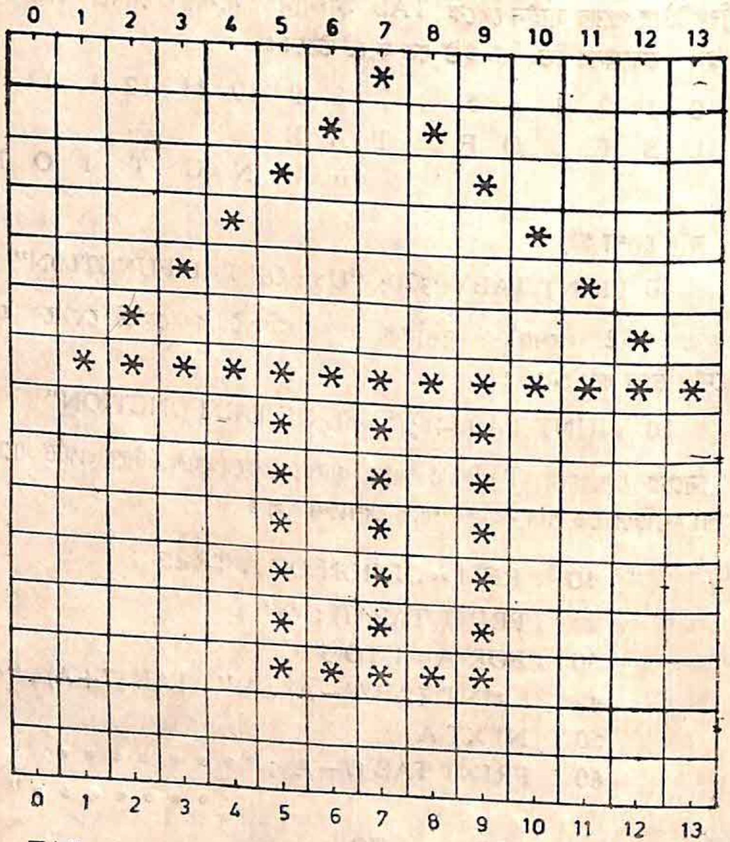
নিচের প্রোগ্রামে TAB ফাংশান ব্যবহার করে কম্পিউটারে একটি গাছের নকশা ফুটিয়ে তোলার সহজ পদ্ধতি দেখানো হল ঃ

|49|

```
10  REM ** DESIGN OF A TREE
20  PRINT TAB (7) ; "*"
30  FOR A=1 TO 5
40  PRINT TAB (7-A) ; "*" ; TAB (7+A) ; "*"
50  NEXT A
60  PRINT TAB (7-A) ; "* * * * * *
                       * * * * * * *"
70  FOR A=1 TO 5
80  PRINT TAB (5) ; "*" ; TAB (7) ; "*" ;
                              TAB (9) ; "*"
90  NEXT A
100 PRINT TAB (5) ; "* * * * *"
110 END
```

এই প্রোগ্রামটি RUN করালে পরের পৃষ্ঠার নকশাটি ভি. ডি. ইউ.-তে ফুটে উঠবে।

RUN



TAB ফাংশানের দ্বিতীয় উদাহরণে আমরা একটা লেখচিত্র আঁকব।

ধরা যাক, একটা গাড়ি ঘণ্টায় 10 কিলোমিটার গতিবেগে ছুটে যাচ্ছে। এই অবস্থায় গাড়িটা ব্রেক কষল। ব্রেক কষার পর 1 মিটার বা 0·01 কিলোমিটার এগিয়ে তার পর গাড়িটা পুরোপুরি থামল। ব্রেক কষার পর কোনও গাড়ি যে-দূরত্ব অতিক্রম করে সেটা D কিলোমিটার মনে করি। ব্রেক কষার সময়ে গাড়ির গতিবেগ যদি ঘণ্টায় V কিলোমিটার হয়, তাহলে একটি সহজ গাণিতিক সূত্রের সাহায্যে D ও V-এর সম্পর্ককে প্রকাশ করা যায়। সেটা হল ঃ

$$D = 0{\cdot}01 \times V^2$$

বেসিকের ভাষায় এই সূত্রটি দাঁড়াবে ঃ

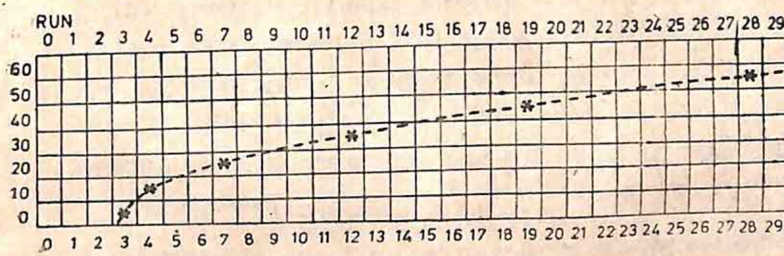
```
10  LET  D=0·01  *  V ↑ 2·0
```

এবারে আমরা V ও D-এর সম্পর্ককে একটি লেখচিত্রের সাহায্যে প্রকাশ করব, এবং বলাবাহুল্য সেই লেখচিত্রটি TAB ফাংশান ব্যবহার করে কম্পিউটার নিজেই এঁকে দেবে। নিচে লেখচিত্র আঁকার প্রোগ্রামটি দেওয়া হল ঃ

50

```
10 REM •• DRAW GRAPH USING TAB FUNCION
20 FOR V = 50 TO 0 STEP −10
30 LET D = 0·01 * V ↑ 2·0
40 PRINT V ; TAB (D+3) ; "*"
50 NEXT V
60 END
```

এই প্রোগ্রামটি RUN করালে V-এর বিভিন্ন মানের জন্য (0, 10, 20, ··· 50) D-এর বিভিন্ন মান (0, 1, 4, 9, 16, 25) পাওয়া যাবে এবং 40 নম্বর নির্দেশ অনুসারে ভি. ডি. ইউ.-এর 3, 4, 7, 12, 19 ও 28 নম্বর অবস্থানে- ছ'টি বিভিন্ন লাইনে ছ'টি * চিহ্ন ফুটে উঠবে। এই তারকাচিত্রগুলো দিয়ে একটি রেখা এঁকে নিলে সেটিই হবে D ও V-এর সম্পর্কের লেখচিত্র। নিচে 50 নম্বর প্রোগ্রামের ফলাফল দেখানো হল ঃ

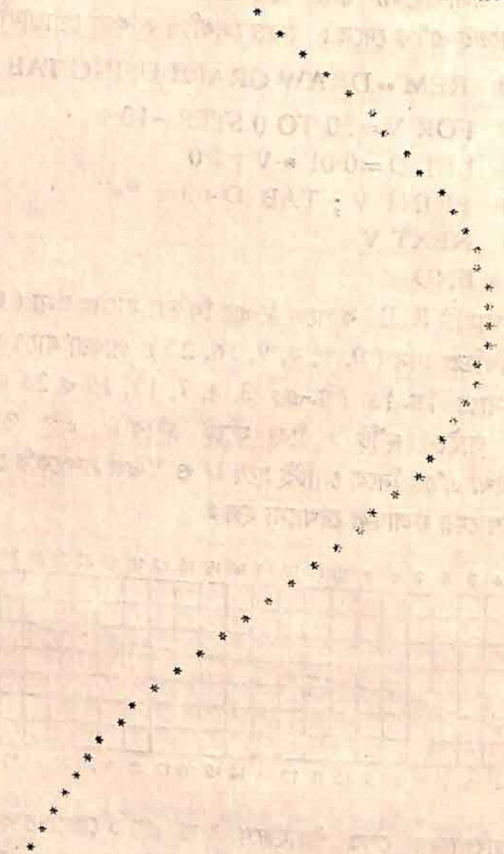


TAB ফাংশানের শেষ উদাহরণে আর একটি লেখচিত্র আঁকার প্রোগ্রাম দেওয়া হল। এই প্রোগ্রামের সাহায্যে 0° থেকে 360° পর্যন্ত (অর্থাৎ, 0 থেকে 2π বা 6·28 রেডিয়ান পর্যন্ত) SIN ফাংশানের বিভিন্ন মান নির্ণয় করা হচ্ছে এবং তা থেকে আঁকা হচ্ছে লেখচিত্র।

51

```
10 REM •• SIN CURVE
20 CLEAR
30 FOR X = 0 TO 6·28 STEP 0·1
40 LET Y = SIN (X)
50 LET A = Y * 20
60 PRINT TAB (30+A) ; "*"
70 NEXT X
80 END
```

এখানে 30 নম্বর নির্দেশে X-এর মান কম্পিউটারকে রেডিয়ানে জানাতে হবে। প্রোগ্রামটি RUN করালে যে উত্তর পাওয়া যাবে তা নিচে দেখানো হল ঃ



লাইব্রেরি ফাংশানের আলোচনা শেষ করে আমরা এবারে চলে যাব শেষ পরিচ্ছেদে। শেষ পরিচ্ছেদে আমরা বেসিক ভাষার কয়েকটি গুরুত্বপূর্ণ সিস্টেম কমান্ডের সঙ্গে পরিচিত হব।

৭ □ বেসিক ঃ সিস্টেম কমাণ্ড

চতুর্থ অধ্যায়ে আমরা প্রোগ্রাম লেখার কাজ শুরু করেছি। কিন্তু প্রোগ্রাম লিখতে গেলে ভুল-ভ্রান্তি হবেই, আর দরকার মতো তা সংশোধন করাও প্রয়োজন। তাছাড়া নতুন একটি পার্সোনাল কম্পিউটারে বসে কাজই বা শুরু করব কেমন করে ? সঠিকভাবে প্রোগ্রাম লিখে তা কম্পিউটারে চালাতে হলে এগুলো জানা প্রয়োজন। সুতরাং পাঠককে চার নম্বর অধ্যায়ের সঙ্গে সঙ্গে এই অধ্যায়টিও ব্যবহার করতে হবে। তবেই সঠিক প্রোগ্রামিং-এর কলাকৌশল তার রপ্ত হবে।

IBM কোম্পানির পার্সোনাল কম্পিউটার এখন সবচেয়ে বেশি জনপ্রিয়। তাই অন্য অনেক কোম্পানিই IBM পার্সোনাল কম্পিউটারের ধাঁচে তাদের কম্পিউটার তৈরি করছেন। এই ধরনের কম্পিউটারগুলোকে বলা হয় 'IBM ক্লোন' বা 'IBM কমপ্যাটিব্‌ল্‌'। অতএব IBM কমপ্যাটিব্‌ল্‌ একটি পার্সোনাল কম্পিউটারে বসে কীভাবে কাজ শুরু করতে হবে সেটাই প্রথমে আলোচনা করা যাক।

কম্পিউটারে প্রোগ্রাম শুরু করার সাধারণ নিয়ম

আমরা সকলেই জানি, কম্পিউটার ইলেকট্রনিক যন্ত্রাংশ দিয়ে তৈরি। সুতরাং এর সাপ্লাই ভোল্টেজ নির্দিষ্ট মানের তুলনায় কম-বেশি হলে ক্ষতির সম্ভাবনা আছে। তাই কম্পিউটারের সাপ্লাই ভোল্টেজের ব্যবস্থা সবসময়েই ভোল্টেজ স্টেবিলাইজারের মারফৎ করতে হবে। ভোল্টেজ স্টেবিলাইজারের কাজ হল মেইন্‌স্‌ ভোল্টেজকে (240 ভোল্ট) স্থির মানে বেঁধে রাখা। ফলে মেইন্‌স্‌ প্লাগের সুইচ অন্‌ করার পর স্টেবিলাইজার মারফৎ কম্পিউটার অন্‌ হবে।

একটা পার্সোনাল কম্পিউটারের মূলত চারটি অংশ ঃ কী-বোর্ড, সিস্টেম ইউনিট, ভি. ডি. ইউ. এবং প্রিন্টার (43 পৃষ্ঠার ছবিতে ভি. ডি. ইউ. যার ওপরে বসানো আছে সেটাই হল সিস্টেম ইউনিট)। মেইন্‌স্‌ প্লাগের সুইচ অন্‌ করার পর ভি. ডি. ইউ.-এর সুইচ অন্‌ করতে হবে, এবং তারপর সিস্টেম ইউনিটের সুইচ অন্‌ করতে হবে। আর যখন কোনও কিছু কাগজে ছাপানোর

প্রয়োজন হবে তখন ভি. ডি. ইউ.-এর সঙ্গে সঙ্গে অন্ করতে হবে প্রিণ্টারের সুইচ।

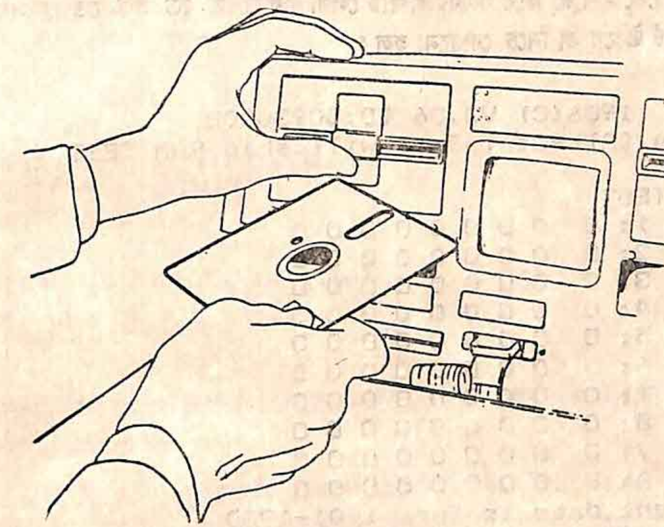
কোনও পার্সোনাল কম্পিউটার কীভাবে কাজ করবে তার জন্য নির্দিষ্ট কয়েকটি প্রোগ্রাম তৈরি করা থাকে। এই প্রোগ্রামগুলো কম্পিউটার ব্যবহারকারীকে সাহায্য করে। কী করে তথ্য ও নির্দেশ ফ্লপি ডিস্ক, কম্পিউটারের স্মৃতিভাণ্ডার ও ভি. ডি. ইউ.-এর মধ্যে প্রয়োজন মতো আদান-প্রদান করতে হবে তারই সুবিন্যস্ত নির্দেশ দেওয়া থাকে এই প্রোগ্রামগুলোর মাধ্যমে। এই প্রোগ্রামগুলোকে সম্মিলিত ভাবে বলা হয় 'ডিস্ক অপারেটিং সিস্টেম' বা সংক্ষেপে 'ডস' (DOS)। 'ডস' প্রোগ্রাম ছাড়া যে-কোনও কম্পিউটার অচল হয়ে পড়বে। কম্পিউটার নির্মাতাদের কাছ থেকে যখন কোনও কম্পিউটার কেনা হবে তখন তারা প্রয়োজনীয় সফটওয়্যারের ফ্লপি ডিস্কগুলো সঙ্গে দিয়ে দেবে। যে ফ্লপি ডিস্কে 'ডস' প্রোগ্রাম রয়েছে তাকেই বলা হয় 'ডস' ডিস্কেট।

পার্সোনাল কম্পিউটারে ফ্লপি ডিস্ক ঢোকানোর জন্য নির্দিষ্ট ফাঁক থাকে। এর চেহারা অনেকটা ডাক বাক্সে চিঠি ফেলার গর্তের মতো। একেই বলা হয় ডিস্ক ড্রাইভ। কোনও কম্পিউটারে একটি অথবা দুটি ফ্লপি ডিস্ক ড্রাইভ থাকে। এই ড্রাইভগুলোর নির্দিষ্ট নাম আছে : A এবং B। কম্পিউটারে একটি ডিস্ক ড্রাইভ থাকলে তার নাম হবে 'ড্রাইভ-এ (A)'। আর দুটি ড্রাইভ থাকলে তাদের নাম রাখা হয় যথাক্রমে 'ড্রাইভ-এ(A)' এবং 'ড্রাইভ-বি (B)'। যদি দুটি ফ্লপি ড্রাইভ ছাড়াও কম্পিউটারে হার্ড ডিস্ক থাকে, তাহলে তার নাম দেওয়া হয় 'ড্রাইভ-সি (C)'। ড্রাইভে ডিস্ক ঢোকানোর পর একটি চাবি ঘুরিয়ে সেটি লক করার ব্যবস্থা থাকে। কম্পিউটার কেনার সময়ে সঙ্গে যে ম্যানুয়াল পাওয়া যায় তা থেকে এই ড্রাইভগুলোর বিষয়ে ভাল করে আগে জেনে নেওয়া দরকার।

কম্পিউটারের সুইচ অন্ করার আগে 'ডস' ডিস্কেটটি ঠিকভাবে ড্রাইভ-এ-তে ঢুকিয়ে দিতে হবে (কম্পিউটারে হার্ড ডিস্ক থাকলে অবশ্য এর আর প্রয়োজন নেই। কারণ, সেক্ষেত্রে 'ডস' প্রোগ্রামগুলো হার্ড ডিস্কেই সঞ্চিত থাকে) এবং সেটিকে লক করে দিতে হবে : ড্রাইভে ডিস্ক ঢোকানোর সময়ে লক্ষ্য রাখতে হবে, ডিস্কের জ্যাকেটের ক্যাপসুল আকারের ছিদ্রটি যেন ওপরের দিকে থাকে এবং ড্রাইভের ভেতরে আগে প্রবেশ করে।

ডিস্ক ঢুকিয়ে কম্পিউটার অন্ করলে কম্পিউটার 'ডস' প্রোগ্রামগুলো ডিস্ক থেকে 'পড়ে' তুলে নেয় নিজের স্মৃতিভাণ্ডারে। এই কাজের সময়ে ডিস্কটি ড্রাইভের ভেতরে ঘোরে এবং সিস্টেম ইউনিটের ভেতরে অবস্থিত রিড-রাইট হেড ডিস্কের ক্যাপসুল আকারের অংশ থেকে (41 পৃষ্ঠার ছবি দ্রষ্টব্য) যাবতীয়

'পড়ার' কাজ করে। যতক্ষণ এই কাজ চলে ততক্ষণ ড্রাইভের পাশে লাগানো একটি ছোট্ট এল. ই. ডি. বাতি জ্বলতে থাকে।



পার্সোনাল কম্পিউটারে ফ্লপি ডিস্ক ঢোকানো হচ্ছে

নিজস্ব স্মৃতিভাণ্ডার পরীক্ষা করার জন্য কম্পিউটারে ব্যবস্থা থাকে। এই পরীক্ষার সাহায্যে কম্পিউটার বুঝতে পারে যে তার স্মৃতিভাণ্ডারে কোনও ত্রুটি দেখা দেয়নি। 'ডস' প্রোগ্রামকে স্মৃতিভাণ্ডারে সঞ্চিত করে নেবার সময় কম্পিউটার স্মৃতিভাণ্ডার পরীক্ষার কাজ চালায় এবং তখন ভি. ডি. ইউ.-তে RAM TEST কথাটি ফুটে ওঠে। RAM TEST-এর কাজ সম্পূর্ণ হলে কম্পিউটার তারিখ জানাতে বলে এবং সময় জানাতে বলে। তারিখ ও সময় জানানোর পর (এই দুটি তথ্যই কী-বোর্ডের বোতাম টিপে জানাতে হবে এবং তারপর রিটার্ন বোতাম টিপতে হবে) 'ডস' প্রম্প্ট্ ফুটে ওঠে ভি. ডি. ইউ.-তে। এর চেহারা নিচে দেখানো হল ঃ

A>=

'ডস' প্রম্প্ট্ ফুটে ওঠার পর কার্সর তার ঠিক পাশেই দপ দপ করে জ্বলতে থাকে। ওপরে কার্সরকেই '=' চিহ্ন দিয়ে দেখানো হয়েছে।

এখন আমাদের কাজ হল, কম্পিউটারকে বোঝানো যে আমরা এখন বেসিক প্রোগ্রাম করতে চাই। এই কাজের জন্য 'ডস' ডিস্কেটটি 'ড্রাইভ-এ' থেকে বের করে নিয়ে বেসিকের জন্য নির্দিষ্ট ডিস্কটি 'ড্রাইভ-এ'-তে সঠিকভাবে ঢোকাতে

হবে এবং সেটিকে লক করে দিতে হবে। এবারে 'ডস' প্রম্প্টের ঠিক পাশেই basic কথাটি টাইপ করে রিটার্ন বোতাম টিপতে হবে।

প্রথম থেকে শুরু করে basic শব্দটি লেখা পর্যন্ত ভি. ডি. ইউ.-তে যে-সব লেখা ফুটে উঠবে তা নিচে দেখানো হল ঃ

```
        1986(C) V3.86 ID:50930008
Press [Alt-Ctrl-Del] will skip RAM TEST

RAM TEST
Bank 1: 0  0 0 0 0 0 0 0 0
Bank 2: 0  0 0 0 0 0 0 0 0
Bank 3: 0  0 0 0 0 0 0 0 0
Bank 4: 0  0 0 0 0 0 0 0 0
Bank 5: 0  0 0 0 0 0 0 0 0
Bank 6: 0  0 0 0 0 0 0 0 0
Bank 7: 0  0 0 0 0 0 0 0 0
Bank 8: 0  0 0 0 0 0 0 0 0
Bank 9: 0  0 0 0 0 0 0 0 0
Bank A: 0  0 0 0 0 0 0 0 0
Current date is Tue  1-01-1980
Enter new date (mm-dd-yy): 10-25-1988
Current time is  0:00:44.54
Enter new time: 10:15:35.00

The ERGO Macro-88(R) MS-DOS v3.20
          (C)Copyright Microsoft Corp. 1981-1986

A>basic




1LIST  2RUN  3LOAD"  4SAVE"  5CONT 6,"LPT1 7TRON 8TROFF9KEY   0SCREEN
```

আগেই আমরা বলেছি বেসিক ভাষার নানারকম সংস্করণ আছে। এইরকম উন্নত সংস্করণগুলোর নাম বিভিন্ন হতে পারে। যেমন IDM-PC-এর ক্ষেত্রে দু-রকম সংস্করণ আছে ঃ BASICA এবং GWBASIC। সুতরাং, এই পার্সোনাল কম্পিউটারে 'ডস' প্রম্প্টের পর আমাদের এই দুটি শব্দের যে-কোনও একটি টাইপ করতে হবে। ধরা যাক, আমরা টাইপ করলাম GWBASIC (অথবা, ছোট হাতের অক্ষরে gwbasic) এবং রিটার্ন বোতাম টিপলাম। তাহলে যে-সব লেখা ভি. ডি. ইউ.-তে ফুটে উঠবে তা দেখানো হল ঃ

```
GW-BASIC 3.20
(C) Copyright Microsoft 1983,1984,1985,1986
60332 Bytes free
Ok
10    REM  SIN  CURVE
20    CLEAR
30    FOR X = 0 TO 6.28 STEP 0.1
40    LET Y = SIN(X)
50    LET A = Y*20
60    PRINT TAB(30+A);"*",
70    LPRINT TAB(30+A);"*",
80    NEXT X
90    END
```

ছবিতে দেখা যাচ্ছে, আমরা বেসিক ভাষায় একটি প্রোগ্রামও লিখেছি (51 নম্বর প্রোগ্রামটি)।

'ডস' প্রম্প্‌টের পাশে GWBASIC লিখে রিটার্ন বোতাম টেপার পর ছবির প্রোগ্রামটুকু বাদে (10 নম্বর নির্দেশ থেকে 90 নম্বর নির্দেশ পর্যন্ত) বাকি সব লেখাই ভি.ডি.ইউ-তে ফুটে উঠবে। ছবির 'Ok' শব্দটি হচ্ছে বেসিক প্রোগ্রাম লেখার প্রম্প্‌ট্‌। এর অর্থ, Ok শব্দটি প্রোগ্রামারকে বুঝিয়ে দিচ্ছে কম্পিউটার এখন বেসিক ভাষায় নির্দেশ গ্রহণ করার জন্য প্রস্তুত। চতুর্থ লাইনে Ok শব্দটি ফুটে ওঠার পরই 51 নম্বর প্রোগ্রামটি কী-বোর্ডের বোতাম টিপে লেখা হয়েছে।

সুতরাং, কোনও কম্পিউটারে বেসিক প্রোগ্রাম লেখার কাজ শুরু করতে হলে আগে জানতে হবে বেসিক ভাষার জন্য কোন্‌ শব্দ (BASIC, BASICA, GWBASIC না অন্য কোনও শব্দ) সেই কম্পিউটারে নির্দিষ্ট করা আছে। এর জন্য কম্পিউটার ম্যানুয়ালটি অবশ্যই ভাল করে পড়ে দেখতে হবে।

এ-পর্যন্ত যে যে নিয়মের কথা আমরা বললাম, পাঠকের সুবিধের জন্য সেগুলোকে সংক্ষেপে ক্রমানুসারে সাজিয়ে দিলাম ঃ

১। 'ডস' ডিস্কেটটি 'ড্রাইভ-এ'-তে সঠিকভাবে ঢুকিয়ে লক কর।
২। স্টেবিলাইজার অন্‌ আছে কিনা দেখে নাও, তারপর মেইন্‌স প্লাগের সুইচ অন্‌ কর।
৩। প্রিন্টার ও ভি. ডি. ইউ.-এর সুইচ অন্‌ করে সবার শেষে কম্পিউটারের, অর্থাৎ সিস্টেম ইউনিটের, সুইচ অন্‌ কর।
৪। RAM TEST সম্পূর্ণ হওয়ার জন্য অপেক্ষা কর। তার পরে

কম্পিউটারের নির্দেশ অনুযায়ী কী-বোর্ডের বোতাম টিপে তারিখ জানিয়ে রিটার্ন বোতাম টেপ।

৫। এবারে কম্পিউটার সময় জানাতে বলবে। সঠিকভাবে সময় জানাও এবং রিটার্ন বোতাম টেপ।

৬। ভি. ডি. ইউ.-তে 'ডস' প্রম্প্‌ট্‌ ফুটে উঠবে। তখন 'ডস' ডিস্কেটটি 'ড্রাইভ-এ' থেকে বের করে নিয়ে বেসিকের জন্য নির্দিষ্ট ডিস্কেট 'ড্রাইভ-এ'-তে ঠিকভাবে ঢুকিয়ে দাও।

৭। এবারে ম্যানুয়াল দেখে বেসিক প্রোগ্রামের জন্য নির্দিষ্ট সঠিক শব্দটি জেনে নাও (BASIC, BASICA অথবা GWBASIC, কিংবা অন্য কোনও শব্দ)। 'ডস' প্রম্প্‌টের পাশেই সেটা টাইপ করে রিটার্ন বোতাম টেপ।

৮। বেসিক প্রোগ্রাম লেখার জন্য নির্দিষ্ট প্রম্প্‌ট্‌ Ok (অথবা, READY, বা অন্য কোনও শব্দ। কম্পিউটার ম্যানুয়ালে এই শব্দটির উল্লেখ থাকবে) ফুটে উঠলে বেসিক প্রোগ্রাম লেখা শুরু কর।

৯। প্রোগ্রাম লেখার কাজ শেষ হলে তার নিচে RUN শব্দটি লিখে রিটার্ন বোতাম টেপ। তাহলেই প্রোগ্রামের উত্তর পাওয়া যাবে। প্রোগ্রামে ভুল থাকলে উত্তরের বদলে কম্পিউটার জানিয়ে দেবে প্রোগ্রামের কোথায় কী ধরনের ভুল আছে। তখন সেই সূত্র ধরে প্রোগ্রাম সংশোধন করে সেই সংশোধিত প্রোগ্রাম আবার RUN করাতে হবে। প্রোগ্রাম নির্ভুল হলে তবেই উত্তর পাওয়া যাবে।

ওপরের ধাপগুলো অনুসরণ করার সময়ে কেউ ইচ্ছে করলে তৃতীয় ধাপের পর চতুর্থ ও পঞ্চম ধাপ এড়িয়ে সরাসরি ষষ্ঠ ধাপের 'ডস' প্রম্প্‌টে পৌঁছে যেতে পারে। এর জন্য কম্পিউটারকে তারিখ ও সময় না জানিয়ে শুধু পর পর দুবার রিটার্ন বোতামটি টিপতে হবে। তবে ওপরের প্রতিটি ধাপ অনুসরণ করার অভ্যেস রাখা ভাল।

সিস্টেম কমাণ্ড

বেসিক ভাষায় প্রোগ্রাম লেখার কাজ শুরু করলেই আমাদের বিভিন্ন সিস্টেম কমাণ্ড ব্যবহার করার প্রয়োজন হয়ে পড়বে। এই কমাণ্ডগুলো ছাড়া প্রোগ্রামিং-এর কাজ একেবারে অচল। আমরা এখন কয়েকটি গুরুত্বপূর্ণ সিস্টেম কমাণ্ডের সঙ্গে পরিচিত হব এবং দেখব কীভাবে এই কমাণ্ডগুলো আমাদের প্রোগ্রামিং-এর কাজে সাহায্য করে।

NEW

কম্পিউটারের মূল স্মৃতির র‍্যাম অংশে প্রোগ্রাম লেখা, মোছা বা সংশোধনের কাজ করা হয়। র‍্যাম-এ একসঙ্গে একটি মাত্র প্রোগ্রাম থাকতে পারে। সুতরাং নতুন কোনও প্রোগ্রাম লেখার কাজ শুরু করার সময়ে প্রথমেই র‍্যামকে 'মুছে' পরিষ্কার করে নেওয়া দরকার। এর ফলে র‍্যামে সঞ্চিত পুরনো কোনও প্রোগ্রাম নতুন কোনও প্রোগ্রামের সঙ্গে এলোমেলো ভাবে মিশে যাবে না। র‍্যাম মোছার কাজ করে NEW নির্দেশটি। নতুন কোনও প্রোগ্রাম লেখার কাজ শুরু করার সময়ে প্রথমেই NEW নির্দেশটি টাইপ করে রিটার্ন বোতাম টিপতে হবে। তাহলেই পুরোনো প্রোগ্রামের যাবতীয় নির্দেশ ও তথ্য কম্পিউটারের স্মৃতি থেকে মুছে যাবে। যদি র‍্যামে একটি প্রোগ্রাম সঞ্চিত হওয়ার পর সেটিকে না মুছে আমরা ধরে রাখতে চাই, তাহলে কম্পিউটারে নতুন কোনও প্রোগ্রাম লেখার আগে পুরনো প্রোগ্রামটিকে সাহায্যকারী স্মৃতিতে সঞ্চয় করে নিতে হবে। পার্সোনাল কম্পিউটারের ক্ষেত্রে ফ্লপি ডিস্কে প্রোগ্রামটি সঞ্চয় করে রাখা যেতে পারে। তারপর NEW নির্দেশটি দিয়ে সেই প্রোগ্রামটি আমরা র‍্যাম থেকে সম্পূর্ণ মুছে ফেলে নতুন প্রোগ্রাম লেখার কাজ শুরু করব।

NEW নির্দেশ ব্যবহার করার সময়ে সতর্ক থাকতে হবে। পুরনো প্রোগ্রামটা যে আমরা মুছে ফেলতে চাই সে-বিষয়ে নিশ্চিত হয়ে তবেই NEW নির্দেশ দিতে হবে।

NEW নির্দেশ ব্যবহার না করলে পুরনো ও নতুন প্রোগ্রাম কীভাবে মিশে যেতে পারে তা খুব সহজ একটি উদাহরণের সাহায্যে দেখানো হল ঃ

```
Ok
10  LET A=5
20  PRINT A
30  END
RUN
  5
Ok
15  LET  B=−4
25  PRINT  B
30  END
RUN
  5
−4
Ok
```

```
NEW
Ok
10  LET  B=−21
20  PRINT  B
30  END
RUN
−21
Ok
```

প্রথম প্রোগ্রামটির পর NEW নির্দেশ না দিয়ে দ্বিতীয় প্রোগ্রাম লেখার ফলে প্রথম ও দ্বিতীয় প্রোগ্রাম মিশে গিয়ে যে-প্রোগ্রামটি তৈরি হয়েছে তা হল ঃ

```
Ok
10  LET  A=5
15  LET  B=−4
20  PRINT  A
25  PRINT  B
30  END
```

সুতরাং দ্বিতীয় প্রোগ্রামটি RUN করানোর সময়ে আমরা আসলে ওপরের মিশ্র প্রোগ্রামটি RUN করিয়েছি এবং উত্তর হিসেবে 5 ও −4 পেয়েছি। অথচ আমরা উত্তর হিসেবে শুধুই −4 চেয়েছিলাম।

কিন্তু তৃতীয় প্রোগ্রাম লেখার শুরুতে আমরা NEW নির্দেশ ব্যবহার করেছি। সুতরাং উত্তরও পেয়েছি নির্ভুল −21, আর এক্ষেত্রে পুরনো প্রোগ্রাম নতুনের সঙ্গে মিশে যায়নি।

কম্পিউটারে প্রোগ্রাম লেখার সময়ে লাইন নম্বর অনুযায়ী পর পর নির্দেশ লেখার কোনও বাধ্য-বাধকতা নেই। প্রোগ্রামার খুশি মতো বিভিন্ন লাইন নম্বরের নির্দেশ টাইপ করলেও কম্পিউটার নিজে সেগুলোকে লাইন নম্বরের ক্রমানুসারে সাজিয়ে নেয়। যেমন, প্রথমে 30 নম্বর লাইন টাইপ করে পরে 10 ও 20 নম্বর লাইন লেখা যেতে পারে।

যদি একই লাইন নম্বর দিয়ে দুটি লাইন লেখা হয়, তাহলে পরে যে-লাইনটি লেখা হবে সেটাই র‍্যামে সঞ্চিত হবে এবং আগের লাইনটি মুছে যাবে। এর ফলে কোনও লাইন নম্বরের নির্দেশ সম্পূর্ণ পাল্টাতে হলে একই লাইন নম্বর আবার টাইপ করে নতুন নির্দেশটি টাইপ করে দিলেই হল। তখন ওই লাইন নম্বরে নতুন নির্দেশটিই সঞ্চিত হবে।

EDIT

কম্পিউটারে কোনও প্রোগ্রাম লেখার পর তা সংশোধনের প্রয়োজন হতে পারে। প্রোগ্রামের ভুল-ত্রুটি শোধরানোর কাজ করতে EDIT নির্দেশ অপরিহার্য।

কোনও প্রোগ্রাম সংশোধনের কাজ সাধারণত তিন রকমের হতে পারে ঃ

১। কোনও লাইনে কম-বেশি পরিবর্তন করা।

২। নতুন কোনও লাইন যোগ করা।

৩। কোনও লাইন সম্পূর্ণ বাদ দেওয়া।

মনে করি, আমরা একটা প্রোগ্রাম লিখলাম এইরকম ঃ

```
10  LET  A=4
20  LET  B=5
30  PRINT A+* B
40  END
```

এই প্রোগ্রামটি চালালে কোনও উত্তর পাওয়া যাবে না। বরং কম্পিউটার জানিয়ে দেবে যে 30 নম্বর লাইনে ভুল হয়েছে। তখন 30 নম্বর লাইনটি শোধরাতে হবে।

30 নম্বর লাইনটা লক্ষ্য করলেই আমরা দেখতে পাব, ভুলবশত + ও * চিহ্ন আমরা পাশাপাশি লিখে ফেলেছি যা সম্পূর্ণ অর্থহীন। যদি আমরা লিখতে চেয়ে থাকি A* B, তাহলে ওই লাইন থেকে বাড়তি '+' চিহ্নটাকে বাদ দিতে হবে। এই কাজ করতে হলে আমরা কম্পিউটারে লিখব ঃ

```
EDIT  30
```

এই কথা লিখে রিটার্ন বোতাম টিপলেই ভি. ডি. ইউ.-তে ফুটে উঠবে 30 নম্বর লাইনটি। তখন কার্সরকে ওই লাইনের '+' চিহ্নের জায়গায় নিয়ে গিয়ে কী-বোর্ডের DEL লেখা বোতামটি টিপতে হবে। তাহলেই '+' চিহ্ন মুছে যাবে এবং তার ডান দিকের অক্ষর বা চিহ্নগুলো বাঁ দিকে এক ঘর করে সরে এসে '+' চিহ্নের শূন্যস্থান পূরণ করে দেবে। তখন সংশোধিত লাইনটি দেখতে হবে এইরকম ঃ

```
30  PRINT A*B
```

এইবার রিটার্ন বোতাম টিপলেই সংশোধিত লাইনটি অশুদ্ধ লাইনের জায়গায় গিয়ে বসবে এবং র‍্যাম থেকে আগের অশুদ্ধ 30 নম্বরের লাইনটি মুছে যাবে। সুতরাং এখন সম্পূর্ণ প্রোগ্রামটি হবে এইরকম ঃ

```
10  LET  A=4
20  LET  B=5
```

```
30  PRINT A*B
40  END
```

এখন যদি আমরা এই প্রোগ্রামে একটা নতুন লাইন যোগ করতে চাই তাহলে প্রথমেই ঠিক করতে হবে সেই লাইনটা আমরা প্রোগ্রামের ঠিক কোন্ জায়গায় লিখতে চাই। ধরা যাক, আমরা 10 ও 20 নম্বর লাইনের মাঝে 'PRINT A' নির্দেশটি ঢোকাতে চাই। এর জন্য নতুন নির্দেশ 'PRINT A'-এর লাইন নম্বর হিসেবে 11 থেকে 19 এর মধ্যে যে-কোনও একটি সংখ্যাকে আমাদের বেছে নিতে হবে। মনে করি, 15 হল সেই সংখ্যা। তাহলে আমরা নিচের লাইনটি কী-বোর্ডে টাইপ করে রিটার্ন বোতাম টিপব :

```
15  PRINT  A
```

তাহলেই প্রোগ্রামটির নতুন চেহারা হবে :

```
10  LET  A=4
15  PRINT A
20  LET  B=5
30  PRINT  A*B
40  END
```

এখন মনে করি, এই প্রোগ্রাম থেকে এক বা একাধিক লাইন আমরা সম্পূর্ণ বাদ দিয়ে দিতে চাই। এর জন্য আমাদের ব্যবহার করতে হবে DELETE নির্দেশ। তাহলেই আমাদের খুশি মতো লাইন আমরা প্রোগ্রাম থেকে সম্পূর্ণ মুছে ফেলতে পারি।

DELETE

DELETE নির্দেশ খুব শক্তিশালী। লাইন বাদ দেবার ব্যাপারে নিশ্চিত হয়ে তবেই DELETE নির্দেশ ব্যবহার করা উচিত।

ধরা যাক, আমরা কোনও প্রোগ্রামের 120 নম্বর লাইনটি সম্পূর্ণ মুছে ফেলতে চাই। তাহলে আমাদের লিখতে হবে :

```
DELETE  120
```

এবারে রিটার্ন বোতাম টিপে দিলেই 120 নম্বর লাইনটি সম্পূর্ণ মুছে যাবে।

যদি আমরা কোনও প্রোগ্রামের শুরু থেকে 120 নম্বর পর্যন্ত সমস্ত লাইন সম্পূর্ণ মুছে ফেলতে চাই, তাহলে টাইপ করতে হবে :

```
DELETE  -120
```

এবং রিটার্ন বোতাম টিপতে হবে।

আবার যদি আমাদের ইচ্ছে হয়, 120 নম্বর লাইন থেকে প্রোগ্রামের শেষ লাইন পর্যন্ত সব নির্দেশ মুছে ফেলব, তাহলে কী-বোর্ডের বোতাম টিপে লিখব ঃ

DELETE 120–

আর তারপর রিটার্ন বোতাম টিপে দেব।

যদি আমরা লিখি ঃ

DELETE 120–190

এবং রিটার্ন বোতাম টিপে দিই তাহলে 120 নম্বর থেকে 190 নম্বর পর্যন্ত সমস্ত লাইনই প্রোগ্রাম থেকে মুছে যাবে।

সুতরাং DELETE নির্দেশ এই চার রকম ভাবে ব্যবহার করা যায়।

কিন্তু যদি কেউ লেখে ঃ

DELETE 120—80

আর তারপর রিটার্ন বোতাম টেপে, তাহলে কম্পিউটার জানিয়ে দেবে ভুল হয়েছে এবং একটিও লাইন সে মুছবে না। কারণ, DELETE নির্দেশে যদি দুটি লাইন নম্বর একসঙ্গে ব্যবহার করা হয় (যেমন, DELETE 120—190), তাহলে বাঁদিকের লাইন নম্বর সব সময় ডান দিকের লাইন নম্বরের চেয়ে কম হতে হবে।

LIST

স্মৃতি থেকে কোনও প্রোগ্রামকে ভি. ডি. ইউ.-তে ফুটিয়ে তোলার জন্য LIST নির্দেশ ব্যবহার করা হয়। প্রোগ্রাম লেখার সময়ে বা তার ভুল শোধরানোর সময়ে LIST নির্দেশ প্রায়ই ব্যবহার করা হয়।

LIST নির্দেশ দিতে হলে DELETE নির্দেশের মতো লাইন নম্বর উল্লেখ করতে হয়। তাহলে উল্লিখিত লাইন নম্বরের নির্দেশগুলো টিভি পর্দায় ফুটে উঠবে। যদি LIST নির্দেশ দিয়ে কোনও লাইন নম্বর উল্লেখ না করে রিটার্ন বোতাম টেপা হয়, তাহলে কম্পিউটারের স্মৃতি থেকে গোটা প্রোগ্রামটাই ভি. ডি. ইউ.-তে ফুটে উঠবে। নিচে কয়েক ধরনের LIST নির্দেশ দেখানো হল ঃ

নির্দেশ	কম্পিউটার কী করবে
LIST	স্মৃতি থেকে গোটা প্রোগ্রামটাই ভি. ডি. ইউ.-তে ফুটিয়ে তুলবে।
LIST 120	স্মৃতি থেকে প্রোগ্রামের শুধু 120 নম্বর লাইনটি ভি. ডি. ইউ.-তে ফুটিয়ে তুলবে।

নির্দেশ	কম্পিউটার কী করবে
LIST–120	প্রোগ্রামের শুরু থেকে 120 নম্বর লাইন পর্যন্ত সব ক'টা লাইন টিভি পর্দায় ফুটিয়ে তুলবে।
LIST 120–	প্রোগ্রামের 120 নম্বর লাইন থেকে শেষ লাইন পর্যন্ত সব ক'টা লাইন টিভি পর্দায় ফুটিয়ে তুলবে।
LIST 120–190	প্রোগ্রামের 120 নম্বর লাইন থেকে 190 নম্বর লাইন পর্যন্ত সব কটা লাইন ভি. ডি. ইউ.-তে ফুটিয়ে তুলবে। এই নির্দেশের ক্ষেত্রে বাঁ দিকের লাইন নম্বর সব-সময়েই ডান দিকের লাইন নম্বরের চেয়ে কম হওয়া চাই। তা না হলে কম্পিউটার নির্দেশ অনুযায়ী কাজ করবে না, এবং একটি লাইনও ভি. ডি. ইউ-তে ফুটিয়ে তুলবে না।

কোনও প্রোগ্রাম চলার সময়ে কখনও LIST নির্দেশ দেওয়া উচিত নয়। আর শুধু LIST লিখে রিটার্ন বোতাম টিপলে কম্পিউটার যেহেতু গোটা প্রোগ্রামটা ফুটিয়ে তোলে, সেহেতু বড় প্রোগ্রামের ক্ষেত্রে এই রকম নির্দেশ ব্যবহার করা উচিত নয়। কারণ তখন প্রোগ্রামের লাইনগুলো ভি. ডি. ইউ.-তে ফুটে উঠে খুব তাড়াতাড়ি পর্দার নিচ থেকে ওপরের দিকে সরে সরে যাবে। ফলে প্রোগ্রাম মোটেই পড়া যাবে না। এরকম অবস্থা হলে কী-বোর্ডের Ctrl এবং Num Lock বোতাম এক সঙ্গে টিপে দিতে হবে। তাহলে পর্দার চলমান লেখা স্থির হবে।

সুতরাং LIST নির্দেশ ব্যবহার করার সময়ে সর্বদাই লাইন নম্বর উল্লেখ করার অভ্যাস থাকা ভাল।

LLIST

LIST নির্দেশের ব্যবহার যেরকম, LLIST-এর ব্যবহারও প্রায় একই রকম। শুধু তফাতের মধ্যে LIST যা করে সবই ভি. ডি. ইউ.-তে, আর LLIST সেই একই কাজ করে প্রিণ্টারে। সুতরাং প্রোগ্রামের প্রতিটি লাইন, বা প্রোগ্রামের কোনও অংশ, কিংবা কোনও একটি বিশেষ লাইন প্রিণ্টারের মাধ্যমে কাগজের ওপরে ছাপাতে চাইলে LLIST নির্দেশ ব্যবহার করতে হবে। LLIST নির্দেশ ব্যবহার করার সময় খেয়াল রাখতে হবে, প্রিণ্টার যেন অন্ করা থাকে।

SAVE

এই নির্দেশ ব্যবহার করে র‍্যাম থেকে ফ্লপি ডিস্কে কোনও প্রোগ্রাম স্থায়ীভাবে সঞ্চিত করে রাখা যায়। যেমন,

SAVE "A : STUDENT"

এই নির্দেশটির অর্থ, 'র‍্যামের বর্তমান প্রোগ্রামটি 'STUDENT' নামে 'ড্রাইভ-এ'-এর ফ্লপি ডিস্কে সঞ্চয় করে রাখ।' SAVE নির্দেশ ব্যবহার করার সময়ে ড্রাইভের নাম—এক্ষেত্রে A—সব সময় উল্লেখ করা উচিত। ড্রাইভের নামের পর অবশ্যই কোলন চিহ্ন দিতে হবে। যেমন, 'A : '।

STUDENT নাম দিয়ে যে-প্রোগ্রামটি ফ্লপি ডিস্কে সঞ্চিত হল, তাকে বলা হবে একটা ফাইল—যে-ফাইলের নাম STUDENT। কম্পিউটারের সাহায্যে যখনই কোনও কিছু ডিস্কে (বা অন্য কোনও সাহায্যকারী স্মৃতিতে) সঞ্চয় করা হয় তখন অবশ্যই তার একটা নাম দিতে হয় (যেমন, ওপরের উদাহরণে STUDENT) এবং সঞ্চিত তথ্যকে বলা হয় একটি ফাইল। যখন আমরা সঞ্চিত কোনও তথ্য বা ফাইল আবার ব্যবহার করতে চাইব, তখন ওই ফাইলের নাম ধরেই তাকে খুঁজতে হবে ফ্লপি ডিস্ক থেকে। এই খোঁজার কাজ করবে কম্পিউটার। কিন্তু ফাইলের কোনও নাম না থাকলে তাকে খুঁজে পাওয়া যাবে না। সেই কারণেই SAVE নির্দেশ দিয়ে কোনও ফাইল ডিস্কে সঞ্চয় করার সময় তার একটা নাম দিতে হয়। নাম না দিলে সেই তথ্য ডিস্কে সঞ্চিত করা যাবে না।

LOAD

LOAD নির্দেশ হল SAVE নির্দেশের অনেকটা বিপরীত। এই নির্দেশ দিয়ে ডিস্কে (অথবা অন্য কোনও সাহায্যকারী স্মৃতিতে) সঞ্চিত কোনও ফাইলকে কম্পিউটারের র‍্যাম-এ তুলে আনা যায়। কারণ র‍্যাম-এ না নিয়ে এলে সেই ফাইলের ওপর কোনওরকম কাজ করা যাবে না। যেমন, STUDENT নামের কোনও প্রোগ্রামকে আমরা হয়ত RUN করাতে চাই। এর জন্য আমাদের নির্দেশ দিতে হবে ঃ

LOAD "A : STUDENT"

এবং তারপর RUN লিখে রিটার্ন বোতাম টিপতে হবে। প্রোগ্রাম চালানো হয়ে গেলে ইচ্ছে করলে আমরা কম্পিউটারের সুইচ অফ করে দিতে পারি। কারণ STUDENT ফাইলটি তখনও ফ্লপি ডিস্কে স্থায়ীভাবে সঞ্চিত রয়েছে।

যদি ডিস্কে সঞ্চিত কোনও প্রোগ্রামকে আমরা সংশোধন বা সংযোজন করে আরও উন্নত করতে চাই, তাহলে প্রথমেই LOAD নির্দেশ ব্যবহার করে

সেই ফাইলটিকে আমরা কম্পিউটারের র‍্যাম-এ তুলে নেব। তারপর LIST ও অন্যান্য নির্দেশ ব্যবহার করে প্রোগ্রামটিকে দরকার মতো সংস্কার করব। এবং সব শেষে SAVE নির্দেশ দিয়ে ফাইলটিকে একই নামে ডিস্কে স্থায়ীভাবে আবার সঞ্চয় করে নেব। এর ফলে প্রোগ্রামটির আগের সংস্করণটি মুছে যাবে, আর ওই নামে নতুন সংস্করণটি সঞ্চিত হয়ে যাবে। এমনটা সম্ভব হল SAVE নির্দেশে ফাইলটির একই নাম (অর্থাৎ, ডিস্কে সঞ্চিত পুরনো নাম) ব্যবহার করার জন্য।

যদি আমরা প্রোগ্রামটির দুটি সংস্করণকেই স্থায়ীভাবে সঞ্চিত করে রাখতে চাই, তাহলে SAVE নির্দেশ দেবার সময় পুরনো নাম ব্যবহার করা চলবে না। নতুন সংস্করণটি নতুন কোনও নামে সঞ্চয় করতে হবে। সেক্ষেত্রে, পরে ইচ্ছে করলে আলাদাভাবে LOAD নির্দেশ ব্যবহার করে দুটি সংস্করণকেই আমরা র‍্যাম-এ তুলে নিয়ে আসতে পারি এবং প্রয়োজনীয় কাজ সারতে পারি।

TRON, TROFF

প্রোগ্রামের ভুলকে অনেক সময় বলা হয় বাগ (BUG), যার অর্থ, ছারপোকা। প্রোগ্রামে কোথাও ভুল হলে সেই ভুল খুঁজে বের করে সংশোধন করতে হয়। সংশোধনের এই কাজকে বলা হয় ডিবাগিং (DEBUGGING)। এমনিতে ডিবাগিং খুব পরিশ্রম ও ধৈর্যের কাজ। এই কাজকে সহজ করে দেবার জন্য কম্পিউটারে কিছু কিছু ব্যবস্থা থাকে। এই ব্যবস্থার মাধ্যমে কম্পিউটার ব্যবহারকারীকে জানিয়ে দেয় কোথায় কী ধরনের ভুল হয়েছে।

যদি কোনও প্রোগ্রাম চালানোর পর কম্পিউটার উত্তর না দেয়, তাহলে সে নিশ্চয়ই ভুলের খবর জানিয়ে দেবে ভি. ডি. ইউ.-এর পর্দায়। কিন্তু অনেক সময় দেখা যায়, কোনও প্রোগ্রাম RUN করালে কম্পিউটার কোনওরকম ভুলের খবর না দিয়ে ঠিক মতোই তা সম্পন্ন করছে, অথচ যে-উত্তর পাওয়া যাচ্ছে তা সঠিক নয়। এরকমটা হতে পারে প্রোগ্রামের কোথাও কোনও চলরাশির মান ভুল দেওয়ার জন্য, কিংবা যুক্তিগত গঠনের কোনও গোলমালের জন্য, অথবা অন্য কোনও জটিল কারণের জন্য। এই ধরনের ভুল খুঁজে বের করা প্রোগ্রামারের পক্ষে বেশ কঠিন। সেই জন্যই তখন শরণ নিতে হয় TRON নির্দেশের। TRON হল TRACE ON-এর সংক্ষিপ্ত রূপ।

ধরা যাক, কম্পিউটারে কোনও প্রোগ্রাম লেখার কাজ শেষ করার পর আমরা TRON কথাটি কী-বোর্ডের সাহায্যে টাইপ করে তারপর প্রোগ্রামটি RUN করালাম। তাহলে প্রতিটি লাইনের নির্দেশ যেমন যেমন ভাবে পালিত হবে, কম্পিউটারের টিভি পর্দায় ঠিক সেই ক্রমানুসারে লাইন নম্বর এবং সেই লাইনের নির্দেশের ফলাফল ফুটে উঠবে। পর্দায় লাইন নম্বরগুলো তৃতীয় বন্ধনীর মধ্যে

ঘেরা থাকবে। এর ফলে আমরা প্রোগ্রামের প্রতিটি ধাপ কী ভাবে পালিত হচ্ছে এবং তার ফলাফল দুই-ই জানতে পারব। সুতরাং কোনও প্রোগ্রামের দুরূহ ভুল সংশোধনের কাজে আমরা TRON নির্দেশ ব্যবহার করতে পারি এবং সন্দেহ নেই, এই নির্দেশের সাহায্যে ভুল খোঁজার কাজ অনেক সহজ হয়ে যাবে।

একটি নির্ভুল প্রোগ্রামকে TRON নির্দেশ ব্যবহার করে RUN করালে ভি. ডি. ইউ.-তে আমরা যা দেখতে পাব তা নিচে দেওয়া হল ঃ

```
Ok
10  LET A=5
20  FOR B=1 TO 3
30  LET A=3*A
40  PRINT A
50  NEXT B
60  END
TRON
Ok
RUN
[10] [20] [30] [40] 15
[50] [30] [40] 45
[50] [30] [40] 135
[50] [60]
Ok
```

ধরা যাক, এর পর আমরা কী-বোর্ড থেকে TROFF টাইপ করে রিটার্ন বোতাম টিপলাম। তাহলে ভি. ডি. ইউ.-তে যা ফুটে উঠবে তা নিচে দেখানো হল ঃ

```
TROFF
Ok
```

TRON নির্দেশ ব্যবহার করে কোনও ভুল প্রোগ্রাম ডিবাগিং-এর কাজ সারা হয়ে গেলে কী-বোর্ডের সাহায্যে TROFF টাইপ করে রিটার্ন বোতাম টিপতে হয়। তাহলেই সেই প্রোগ্রাম আবার RUN করালে তার প্রতিটি লাইন নম্বর ও সেই লাইনের নির্দেশের ফলাফল ভি. ডি. ইউ.-তে আর ফুটে উঠবে না, সরাসরি প্রোগ্রামের ফলাফল ফুটে উঠবে।

যেমন, TROFF নির্দেশ ব্যবহার করার পর ওপরের প্রোগ্রামটি আবার RUN করালে উত্তর পাওয়া যাবে ঃ

```
Ok
RUN
  15
  45
  135
```

অর্থাৎ, TRON নির্দেশকে অকেজো করার জন্যই TROFF নির্দেশ প্রয়োজন। TROFF হল TRACE OFF-এর সংক্ষিপ্ত রূপ।

SYSTEM

বেসিক ভাষায় প্রোগ্রামের কাজ শেষ করার পর কম্পিউটারে অন্য কোনও কাজ করতে চাইলে প্রোগ্রামারকে আবার 'ডস' প্রম্প্‌টে ফিরে আসতে হয়। কারণ কম্পিউটারে যে-কোনও কাজ শুরুর প্রাথমিক ধাপ হচ্ছে 'ডস' প্রম্প্‌ট্‌। 'ডস' প্রম্প্‌টে ফিরে আসতে হলে বেসিক ভাষার SYSTEM নির্দেশ ব্যবহার করতে হবে। SYSTEM নির্দেশ পেলে কম্পিউটার বেসিকের সমস্ত ফাইল বন্ধ করে ফিরে আসবে 'ডস্‌' প্রম্প্‌ট্‌ চিহ্নে।

যে-কটি সিস্টেম কমান্ড এখানে আলোচনা করা হল তা ছাড়াও আরও অনেক সিস্টেম কমান্ড রয়েছে। সেগুলো সম্পর্কে জানতে হলে কম্পিউটারের সঙ্গে পাওয়া ম্যানুয়ালটি ভাল করে দেখতে হবে। এছাড়া এই বইয়ে আলোচিত যাবতীয় নির্দেশ ইত্যাদি হাতে কলমে যাচাই করে দেখার সময়েও প্রয়োজনে কম্পিউটার ম্যানুয়ালের সাহায্য নিতে হবে। কারণ এক কম্পিউটার থেকে অন্য কম্পিউটারে বেসিক ভাষার নির্দেশ ইত্যাদির সামান্য ইতরবিশেষ হতে পারে। তবে এই তফাৎ খুবই নগণ্য।

সব শেষে একটা জরুরী পরামর্শ ঃ কম্পিউটার প্রোগ্রামিং ভাল ভাবে রপ্ত করার একমাত্র পথ হল কী-বোর্ডে বোতাম টেপার পিছনে প্রচুর সময় খরচ করা। অর্থাৎ, কোনও প্রোগ্রাম লিখে সেটা হাতে-কলমে যাচাই করার অভ্যেসটা নিয়মিত রাখতে হবে। সুতরাং কোনও শিক্ষার্থীকে নিত্য-নতুন প্রোগ্রাম লিখতে হবে এবং তা কম্পিউটারে চালিয়ে দেখতে হবে। এর জন্য কোনও 'সংক্ষিপ্ত রাজকীয় পথ' নেই। পরিশ্রম ও অভ্যাসই প্রোগ্রামিং-এ দক্ষতা অর্জনের এক এবং একমাত্র মন্ত্র।

পরিশিষ্ট □ এক

এই বইয়ে আলোচিত বেসিক ভাষার মূল শব্দ / নির্দেশের তালিকা

নাম	পৃষ্ঠা	নাম	পৃষ্ঠা
DATA	71	LET	64
DIM	113	LPRINT	69
END	75	ON-GOSUB	126
FOR—TO—STEP—NEXT	102	ON-GOTO	111
GOSUB	119	PRINT	69
GOTO	87	READ	71
IF—THEN	94	REM	86
IF—THEN—ELSE	98	RESTORE	71
INPUT	68	RETURN	120
		STOP	96

এই বইয়ে আলোচিত বেসিক ভাষার লাইব্রেরি ফাংশানের তালিকা

নাম	পৃষ্ঠা	নাম	পৃষ্ঠা
ABS	128	MID$	130
ACS	133	RIGHT$	117
ASN	133	RND	129
ATN	133	SGN	129
COS	132	SIN	132
COT	132	SQR	128
EXP	130	STR$	117
INT	119	TAB	133
LEFT$	117	TAN	132
LEN	117	VAL	117
LOG	131		

এই বইয়ে আলোচিত বেসিক ভাষার সিস্টেম কমাণ্ড

পরিশিষ্ট □ দুই   সংক্ষিপ্ত কম্পিউটার শব্দকোষ

ACCUMULATOR ঃ অ্যাকুমুলেটর হল সি. পি. ইউ.-এর গণনা বিভাগের মধ্যে একটি বিশেষ জায়গা যেখানে গাণিতিক সমাধান ইত্যাদি সংরক্ষিত থাকে। বিভিন্ন ধরনের সমাধান সঞ্চিত রাখার জায়গা থাকে এই অ্যাকুমুলেটরে। গণনা বিভাগে অঙ্ক চলাকালীন বিভিন্ন তথ্য পর পর জমা হয় এখানে। দরকার মতো এই সব তথ্য বা নির্দেশ অ্যাকুমুলেটর থেকে নিয়ে এসে কম্পিউটার কাজে লাগায়।

ADDRESS ঃ কোনও তথ্য বা নির্দেশ বাইনারি কোডে রূপান্তরিত করে কম্পিউটারের মূল স্মৃতিতে রাখা হয়। এক-একটি স্মৃতিকোঠায় এক-একটি তথ্য থাকে, তাই প্রত্যেকটি স্মৃতিকোঠাকে আলাদা ভাবে শনাক্ত করার ব্যবস্থা থাকা প্রয়োজন। সেই কারণেই স্মৃতির প্রত্যেক স্থানের এক-একটা আলাদা ঠিকানা বা অ্যাড্রেস আছে। যেমন ঠিকমতো চিঠি বিলি করার জন্য বাড়ির ঠিকানা থাকা প্রয়োজন তেমনি কম্পিউটারের সঠিক কাজের জন্য এই 'ঠিকানার' প্রয়োজন। কোনও বিশেষ তথ্য স্মৃতির যে-ঠিকানায় লেখা হবে সেই তথ্য ওই ঠিকানাতেই সঞ্চিত হবে এবং পরে তা পাঠ করার সময় কম্পিউটার ওই ঠিকানার স্মৃতিকোঠা থেকেই তথ্যটি পাঠ করবে।

ALGORITHM ঃ কোনও একটি বিশেষ সমস্যা সমাধানের জন্য কম্পিউটারকে যথাযথভাবে নির্দেশ দিতে হয়। তার আগে আমাদের ওই সমস্যার সমাধান-পদ্ধতি নির্ধারণ করতে হয়। অর্থাৎ, কীভাবে পর পর পর্যায়ক্রমিক ভাবে কাজ করলে সমস্যাটির সমাধান করা যায়। এই সমাধান পদ্ধতির নামই হল অ্যালগরিদম্।

ALPHANUMERIC ঃ তথ্য বা নির্দেশ দু-ধরনের চিহ্ন দিয়ে প্রকাশ করা হয় ঃ বর্ণমালার অক্ষর (ALPHAbet), যেমন A, B, C, D……, এবং সংখ্যা (NUMERIC), যেমন 1, 2, 3, 4……, ইত্যাদি। এই দু-ধরনের চিহ্নকে মিলিতভাবে আলফানিউমারিক বলা হয়।

ANALOG COMPUTER ঃ কোনও সঙ্কেত দুরকমের হতে পারে—ডিজিটাল, অর্থাৎ বাইনারি, অথবা অ্যানালগ। উদাহরণস্বরূপ বলা যেতে পারে টেলিফোনের সিগন্যাল অ্যানালগ, কিন্তু টেলিগ্রাফ প্রেরণ করা হয় মোর্স

কোডের সাহায্যে—অর্থাৎ ড্যাশ (—), ডট (·)-এর মাধ্যমে — যা হল ডিজিটাল সঙ্কেত। বেশির ভাগ কম্পিউটারই ডিজিটাল কম্পিউটার। কিন্তু অ্যানালগ কম্পিউটারও বিজ্ঞান এবং প্রযুক্তিতে ব্যবহার করা হয়। এই কম্পিউটারে সমস্ত সঙ্কেতকে বৈদ্যুতিক ভোল্টেজ বা কারেণ্টে পরিবর্তন করা হয়। কোনও সঙ্কেতের মান তুলনামূলক বৈদ্যুতিক ভোল্টেজ বা কারেণ্টে প্রকাশ করে বৈদ্যুতিক বর্তনীর মাধ্যমে তাদের সমাধানের কাজে লাগানো হয়।

ARITHMETIC AND LOGIC UNIT (ALU) ঃ এটি হল সি. পি. ইউ.-এর গণনা বিভাগ। গণনা বিভাগ যোগ, বিয়োগ, গুণ, ভাগ ছাড়াও দুটি সংখ্যার তুলনা করতে পারে। এছাড়া অ্যারিথমেটিক ও লজিক ইউনিটের কিছু যুক্তিগ্রাহ্য ক্রিয়াকলাপের ক্ষমতাও আছে।

ASCII ঃ অ্যাস্‌কি কোড-এর পুরো নাম হল আমেরিকান স্ট্যান্ডার্ড কোড ফর ইনফরমেশন ইণ্টারচেঞ্জ (American Standard Code for Information Interchange)। এই আন্তর্জাতিক কোড দিয়ে যে-কোনও মডেলের যে-কোনও কম্পিউটারের সঙ্গে ভাব বিনিময় করা যায়। এই কোডে 7 বিট অথবা 8 বিট শব্দ দিয়ে ইংরেজী অক্ষর, সংখ্যা, যতি চিহ্ন, পাটিগণিতীয় চিহ্ন ও অন্যান্য 128-টি সঙ্কেত প্রকাশের ব্যবস্থা আছে। যখনই আমরা কী-বোর্ডে 'E' অক্ষর লেখা বোতামটি টিপলাম তখনই '01100101' বাইনারি সঙ্কেতটি—যার অর্থ ছোট হাতের 'e'—কম্পিউটারে প্রবেশ করল। যদি কী-বোর্ডের Caps Lock বোতাম টিপে তারপর 'E' অক্ষরের বোতাম টিপি—অর্থাৎ, যদি বড় হাতের 'E' চাই—তাহলে '01000101' বাইনারি সঙ্কেতটি কম্পিউটারে প্রবেশ করবে। এক্ষেত্রে বাইনারি সঙ্কেতগুলোই হচ্ছে 'E' অথবা 'e'-এর সমতুল অ্যাসকি কোড।

ASSEMBLER ঃ অ্যাসেম্ব্‌লার কোনও পরিচিত ইংরেজী ভাষার অক্ষরকে মেশিনের ভাষায় রূপান্তরিত করে। যখন উঁচুদরের ভাষায় LOAD, READ, PRINT, END এইসব নির্দেশ আমরা কম্পিউটারকে জানাই তখন অ্যাসেম্ব্‌লার ওই নির্দেশগুলোকে মেশিনের ভাষায় বা যান্ত্রিক ভাষায় রূপান্তরিত করে। তখন ওইসব নির্দেশ কম্পিউটারের বোধগম্য হয়।

BIT ঃ BInary digiT-এর সংক্ষিপ্ত রূপ হল—বিট (BIT)। অর্থাৎ, বিট হল '1' অথবা '0' যে-কোনও একটি বাইনারি সংখ্যা।

BUS ঃ যে-পথ দিয়ে কম্পিউটারের তথ্য বা নির্দেশ যাতায়াত করে তাকে বাস (BUS) বলে। বাসযাত্রীরা যেমন বাসে যাতায়াত করে

তেমনি ডেটা বাসের মাধ্যমে তথ্য আদান-প্রদান হয়। বাস সিস্টেম কম্পিউটার এবং ইনপুট/আউটপুট ব্যবস্থার মধ্যে যোগসূত্র বিশেষ। আবার কম্পিউটারের মধ্যেও, অর্থাৎ সি. পি. ইউ. এবং নিজস্ব স্মৃতির মধ্যেও, তথ্য চলাচলের কাজ হয় বাসের মাধ্যমে। বস্তুত ডেটা বাস বললে কয়েকগুচ্ছ তারকে বোঝায়, যারা কম্পিউটার এবং তার ইনপুট বা আউটপুট ব্যবস্থার সঙ্গে যুক্ত থাকে। বাসগামী তথ্যের গতিপথের সামগ্রিক পরিচালনার দায়িত্ব থাকে নিয়ন্ত্রণ বিভাগের।

BYTE ঃ সাধারণত 8-টি বিট-এর সমষ্টি দিয়ে একটি BYTE তৈরি হয়। তথ্য বা নির্দেশ বাইটের সাহায্যে জানানো হয়। $2^{10} = 1024$ বাইটকে 1 কিলোবাইট (1 KB) বলে। আবার $2^{20} = 1048576$ বাইট হল এক মেগাবাইট (1 MB)। কোনও কম্পিউটারের স্মৃতির ক্ষমতা KB বা MB দিয়ে প্রকাশ করা হয়।

COMPILER ঃ এটা কম্পিউটারের একটা প্রোগ্রাম বা সফ্‌টওয়্যার যা কোনও উঁচুদরের ভাষাকে যান্ত্রিক ভাষায় রূপান্তরিত করে। অর্থাৎ, উঁচুদরের ভাষাকে '0' এবং '1' দিয়ে তৈরি বাইনারি ভাষায় পরিবর্তিত করে কম্পাইলার।

CURSOR ঃ কী-বোর্ডের সাহায্যে কম্পিউটারের টিভি পর্দায় যেখানে লেখা হবে তার অবস্থান নির্ধারণ করার জন্য একটি উজ্জ্বল আলোক-চিহ্ন হল কার্সর। কোনও কোনও কম্পিউটারে এটা একটা ছোট আয়তাকার আলোক চিহ্ন। আবার কোনও কোনও কম্পিউটারে কার্সর একটি উজ্জ্বল ছোট সরলরেখা। কী-বোর্ডের কার্সর কণ্ট্রোল বোতামগুলো টিপে কার্সরকে ওপরে নীচে ডাইনে বাঁয়ে সরানো যায়। কোনও প্রোগ্রাম মোছা, নতুন লাইন লেখা, সংশোধন করা ইত্যাদি কাজে কার্সর অপরিহার্য।

DAISY WHEEL PRINTER : ডেইজি হুইল প্রিণ্টারে বিভিন্ন ধরনের বর্ণ বা চিহ্ন সাজানো থাকে কতগুলি স্পোকের ওপর। ডেইজি হুইল সব সময় ঘুরতে থাকে। কম্পিউটারের নির্দেশ অনুযায়ী যা ছাপতে হবে, তার প্রত্যেকটি বর্ণ অথবা চিহ্নের বাইনারি সঙ্কেত কম্পিউটার প্রিণ্টারে পাঠায়। তখন ডেইজি-প্রিণ্টারে অবস্থিত বর্ণমালার নির্দিষ্ট বর্ণটিকে একটি হাতুড়ি এসে আঘাত করে। সেই আঘাতের ফলে কালি মাখা রিবনের সাহায্যে কাগজে ছাপা হয়ে যায় বর্ণটি। সাধারণত ডেইজি হুইল প্রিণ্টার সেকেণ্ডে 20 থেকে 30 টি বর্ণ ছাপতে পারে। এর ছাপার গতি লাইন প্রিণ্টারের থেকে কম। কিন্তু দামও লাইন প্রিণ্টারের তুলনায় অনেক কম বলে এই প্রিণ্টার যথেষ্ট জনপ্রিয়।

DAS ঃ এর পুরো নাম হল ডেটা অ্যাকুইজিশন সিস্টেম (Data Acquisition System)—অর্থাৎ, কম্পিউটারভিত্তিক তথ্য সংগ্রহ পদ্ধতি। DAS-এর সাহায্যে বিভিন্ন তথ্য সংগ্রহ করা, সুপরিকল্পিতভাবে তথ্য সাজানো, ওই তথ্য দিয়ে বিভিন্ন গণনা করা এবং প্রয়োজন মতো তার বিশ্লেষণ করা হয়। যেমন, একটি স্বয়ংক্রিয় তাপবিদ্যুৎ উৎপাদন কেন্দ্রের কথাই ধরা যাক। সেখানে পুরো কার্যপদ্ধতিটি নিয়ন্ত্রণ করছে কম্পিউটারের সঙ্গে যুক্ত অসংখ্য যন্ত্র। নিয়ন্ত্রণ-যন্ত্রগুলো তাপবিদ্যুৎ কেন্দ্রের কার্যপদ্ধতি থেকে পাওয়া যাবতীয় তথ্য কম্পিউটারকে সরবরাহ করছে। কম্পিউটার সেই তথ্য ঠিকমত গুছিয়ে রাখছে এবং প্রাপ্ত তথ্যের ভিত্তিতে প্রয়োজনীয় গণনা ও বিশ্লেষণ করে নিয়ন্ত্রণ সংক্রান্ত সিদ্ধান্ত গ্রহণ করছে। DAS পদ্ধতিতে উৎপাদন সংক্রান্ত প্রয়োজনীয় বিভিন্ন রিপোর্টও প্রিণ্ট করা হয়। এই সব রিপোর্ট থেকে প্ল্যাণ্টের কোনও ত্রুটি বিশ্লেষণ করা এবং ভবিষ্যৎ পরিকল্পনা রূপায়ণ করা সম্ভব।

DATA BASE : সুসংবদ্ধ তথ্য দিয়ে তৈরি কয়েকটি বিশাল ফাইলের সমষ্টি হল ডেটা বেস। বিভিন্ন প্রয়োগের ওপর ভিত্তি করে ডেটা বেস-এ বিশেষ বিশেষ ফাইল তৈরি করা হয়। একটা ফাইল তৈরি হয় অনেকগুলি তথ্যসমষ্টি বা রেকর্ড দিয়ে। এই রেকর্ডে তথ্যগুলিকে একটি বিশেষ প্রথাসিদ্ধ রীতিতে লেখা হয়। উদাহরণ হিসেবে বলা যায়, একটি জনগণনায় প্রত্যেকের নাম, ঠিকানা, পেশা, বয়স ইত্যাদি একসঙ্গে নিয়ে এক-একটি রেকর্ড তৈরি হতে পারে এবং সেই রেকর্ডগুলোর সমষ্টি হল জনগণনার একটি ফাইল। এই ফাইল নিয়ে কম্পিউটার জনগণনার ওপরে প্রয়োজনীয় বিশ্লেষণ করতে পারবে। জনগণনার সঙ্গে সম্পর্ক আছে এরকম একাধিক ফাইলের সমষ্টি নিয়ে তৈরি হতে পারে জনগণনার একটি ডেটা বেস।

DBMS : এর পুরো কথাটা হল ডেটা বেস ম্যানেজমেণ্ট সিস্টেম (Data Base Management System)। কোনও কম্পিউটারের কার্যকরী ও সুদক্ষ ব্যবহার নির্ভর করছে কত সুচারুভাবে আমরা নির্দিষ্ট সংখ্যক নির্দেশের ওপরে ভিত্তি করে বিভিন্ন সমস্যার সমাধান পদ্ধতিকে কম্পিউটারের বোধগম্য ভাষায় রূপান্তরিত করতে পারি। DBMS হল একটা বিশেষ ধরনের সফ্‌ট্‌ওয়্যার যা দিয়ে আমরা ডেটা বেস তৈরি করতে পারি ও খুশি মতো তা ব্যবহার করতে পারি। ওই সফ্‌ট্‌ওয়্যারের সাহায্যে আমরা প্রয়োজনীয় তথ্য কোনও ফাইল থেকে বের করে দরকার মতো সাজাতে পারি। একটি প্রতিষ্ঠানে সমস্ত কর্মীর মাসিক বেতনের হিসেব এবং সেখানকার আয়-ব্যয় সংক্রান্ত অন্যান্য হিসেব-নিকেশ, মাধ্যমিক পরীক্ষার সমস্ত পরীক্ষার্থীর প্রাপ্ত নম্বরের ট্যাবুলেশন ও মার্কশিট

তৈরি, জনগণনায় প্রাপ্ত বিভিন্ন তথ্যের তালিকা ও হিসেব ইত্যাদিতে কম্পিউটারের ব্যবহার আজ ব্যাপক। DBMS-এর সাহায্যে এই ক্রমবর্ধমান তথ্যের নির্ভুল এবং দ্রুত বিশ্লেষণ করা সম্ভব। ধরা যাক, একটি কোম্পানি কোনও প্রতিষ্ঠানে গত দশ বছরে কী কী জিনিস সরবরাহ করেছে তার একটা তালিকা দরকার। কম্পিউটার DBMS-এর সাহায্যে বিভিন্ন সরবরাহকারীর মধ্যে থেকে সেই বিশেষ সরবরাহকারী কোম্পানিকে পলকে খুঁজে বের করে সঙ্গে সঙ্গে প্রয়োজনীয় তালিকার প্রিণ্ট-আউট দিয়ে দিতে পারে।

DEBUGGING : কম্পিউটারের হার্ডওয়্যার বা সফ্ট্ওয়্যারের ত্রুটি-বিচ্যুতি দূর করার সঠিক পদ্ধতির নাম ডিবাগিং। বিভিন্ন কম্পিউটারের অপারেটিং সিস্টেম-এ ডিবাগ করার নির্দেশ সঞ্চিত থাকে। এর সাহায্যে কম্পিউটার নিজেই কোনও প্রোগ্রাম, স্মৃতিকোষ বা সি. পি. ইউ.-এর ত্রুটি-বিচ্যুতি বিশ্লেষণ করতে পারে এবং সেই ত্রুটি শুধরে নেয় অথবা সেই ত্রুটি সম্পর্কে ব্যবহারকারীকে সঙ্কেত দেয়।

DIAGNOSTICS : প্রতিটি কম্পিউটারেই ডায়াগনোস্টিক রুটিন আছে। এর সাহায্যে কম্পিউটার রুটিনমাফিক ভাবে প্রতিটি হার্ডওয়্যার এবং সফ্ট্ওয়্যারের ত্রুটি-বিচ্যুতির অনুসন্ধান করে তার ভি. ডি. ইউ. পর্দায় জানিয়ে দেয় সেই ত্রুটি কোথায় এবং কী ভাবে তাকে দূর করা যায়।

FIRMWARE : আমরা হার্ডওয়্যার এবং সফ্ট্ওয়্যারের সঙ্গে পরিচিত। ফার্মওয়্যার হল কম্পিউটারের মধ্যে পাকাপাকি ভাবে রক্ষিত কোনও প্রোগ্রাম অথবা নির্দেশ। এটা সাধারণত রিড ওনলি মেমোরি বা রম্ (ROM)-এর সাহায্যে লেখা। এটা হার্ডওয়্যার তো নয়ই, এমনকি সঠিক অর্থে সফ্ট্-ওয়্যারও নয়—কেননা এটা কম্পিউটারের মধ্যে রম্ দিয়ে পাকাপাকি ভাবে সঞ্চিত। ফার্মওয়্যার নানাপ্রকার পৌনঃপুনিক নির্দেশ পালন করে—যেমন, কম্পিউটারের বিভিন্ন বাস-এ তথ্য পরিচালনা করা অথবা কোনও তথ্য বা নির্দেশ টিভি পর্দা বা প্রিণ্টারে পাঠানো ইত্যাদি।

GRAPHICS : গ্রাফিক্সের ব্যবহার কম্পিউটারের সাহায্যে ডিজাইন বা নক্শা আঁকার কাজকে আরও সহজ ও সুন্দর করেছে। বিভিন্ন জ্যামিতিক চিত্র, সরলরেখা, বৃত্ত থেকে শুরু করে বহু রঙিন জটিল চিত্র কম্পিউটার গ্রাফিক ভাষায় সংরক্ষিত করতে পারে এবং প্রয়োজন মতো ব্যবহার করে বিভিন্ন চিত্ররূপ অঙ্কন করতে পারে। গ্রাফিক্স্ তৈরি করতে হলে কম্পিউটারের এমন সফ্ট্ওয়্যার প্রয়োজন যা দিয়ে আমরা ভি. ডি. ইউ.-এর পর্দায় একটা আলোকিত বিন্দুকে চিহ্নিত করতে পারি। এজন্য প্রতিপ্রভ

পর্দাটি ছোট ছোট পিক্সেল (PIXEL) বা পিকচার এলিমেণ্ট-এ ভাগ করা থাকে। পর্দায় একটা পিক্সেল-এর অবস্থান x এবং y অক্ষ অনুযায়ী চিহ্নিত হয়। অনেকগুলো পিক্সেল মিলে একটা নকশা ফুটিয়ে তোলে। সাধারণত বারো ইঞ্চি পর্দায় 160 × 100টি পিক্সেল থাকে—অর্থাৎ, বাঁদিক থেকে ডান দিকে 160 টি আলোকবিন্দু, এবং ওপর থেকে নিচে 100 টি আলোকবিন্দু। সুতরাং মোট 16000 আলোকবিন্দু দিয়ে পর্দাটা তৈরি। কী-বোর্ড, জয়স্টিক, লাইট পেন, ট্র্যাকবল বা মাউস প্রভৃতির সাহায্যে নকশা আঁকা যায়। গ্রাফিক-এর ব্যবহার কম্পিউটারকে এক সুদক্ষ নকশা-আঁকিয়েতে পরিণত করেছে। এই 'শিল্পী'র এমনই দক্ষতা যে শুধুমাত্র বিশেষ একটি নকশাকে আঁকতে পারাটাই তার শেষ কথা নয়—ওই নকশাটিকে বিভিন্ন কোণ থেকে দেখলে কেমন দেখাবে তাও সে আঁকতে পারে।

HANDSHAKING : হ্যাণ্ডশেকিং হল দুটো কম্পিউটারের মধ্যে তথ্য আদান-প্রদানের প্রস্তুতির নির্দেশ। এই নির্দেশের ফলে কম্পিউটার দুটি বুঝতে পারে যে তারা তথ্য আদান-প্রদানের জন্য তৈরি। আবার তথ্য বিনিময় হয়ে যাবার পর এই নির্দেশ জানিয়ে দেয় যে তথ্য আদান-প্রদানের কাজ শেষ হয়েছে। শুধু দুটো কম্পিউটারই নয়, কম্পিউটার ও তার ভি ডি. ইউ. বা প্রিণ্টারের মধ্যেও হ্যান্ডশেকিং হয়।

HARD COPY : ভি. ডি. ইউ.-তে, অর্থাৎ টারমিনালের পর্দায়, যে-ছবি আমরা পাই কাগজের ওপর তারই প্রিণ্ট-আউট হল হার্ড কপি। অনেক সময় কম্পিউটারের সাহায্যে কোনও গ্রাফিক্সের হার্ড কপি করার জন্য প্লটারও ব্যবহার করা হয়। সাধারণত প্রতিটি প্রোগ্রামেরই একটা করে হার্ড কপি রাখা উচিত। যদি কোনও কারণে ফ্লপি ডিস্ক নষ্ট হয়ে যায় বা ত্রুটিযুক্ত হয়ে পড়ে তবে হার্ড কপি দেখে আবার সঠিক প্রোগ্রামটা কম্পিউটারে সঞ্চয় করা যেতে পারে।

HARDWARE : কম্পিউটারের যে-সব ভৌত অংশ আছে—যেমন, সি. পি. ইউ., কী-বোর্ড, প্রিণ্টার, ভি. ডি. ইউ., ইলেকট্রনিক কার্ড, ফ্লপি ডিস্ক—এগুলো হল কম্পিউটারের হার্ডওয়্যার।

IC CHIP : ইণ্টিগ্রেটেড সার্কিট-এর আবিষ্কার ইলেকট্রনিক্সের জগতে আলোড়ন তুলেছিল। তার আগে দশ-বারোটি ট্রানজিস্টার, রেজিস্ট্যান্স, ক্যাপাসিট্যান্স ইত্যাদি দিয়ে তৈরি একটা অ্যামপ্লিফায়ার সার্কিট প্রায় 36 বর্গ ইঞ্চি জায়গা নিত। অর্ধপরিবাহী সিলিকনের একটি পাতলা টুকরো (WAFER) নিয়ে তার ওপরে যদি একটি ইণ্টিগ্রেটেড সার্কিট চিপ তৈরি করা যায় (I.C. CHIP) তাহলে ওই অ্যামপ্লিফায়ার তৈরি করা সম্ভব হবে $1'' \times \frac{1}{4}'' \times \frac{1}{8}''$

জায়গার মধ্যে। ইন্টিগ্রেটেড সার্কিটের বিকাশ এবং অগ্রগতির ফলে বর্তমানে 100 টা ট্রানজিস্টার দিয়ে তৈরি কোনও জটিল বর্তনীকে মাত্র $1\frac{1}{2}'' \times \frac{1}{2}'' \times \frac{1}{8}''$ জায়গার মধ্যে 'বন্দী' করা সম্ভব। আই. সি. চিপ ব্যবহারের ফলে কম্পিউটারের চেহারা ক্রমেই ছোট হয়েছে এবং তার বিদ্যুৎ খরচও গেছে কমে।

INTERPRETER ঃ ইণ্টারপ্রিটার কথাটার অর্থ হল 'অনুবাদক'। কম্পিউটারের ক্ষেত্রে ইণ্টারপ্রিটার হল এমন একটা সফ্‌ট্‌ওয়্যার প্রোগ্রাম যা উঁচুদরের ভাষাকে অনুবাদ করে কম্পিউটারের বোধগম্য যান্ত্রিক ভাষায় রূপান্তরিত করে। কম্পাইলার এবং ইণ্টারপ্রিটারের কাজ মূলত একই ধরনের হলেও ইণ্টারপ্রিটার কোনও নির্দেশকে যান্ত্রিক ভাষায় রূপান্তরিত করার সঙ্গে সঙ্গেই তাকে কার্যকরী করে, কিন্তু কম্পাইলার তা করে না। বেসিক ভাষাকে কম্পিউটারের বোধগম্য বাইনারি ভাষায় রূপান্তরিত করা হয় ইণ্টার-প্রিটারের মাধ্যমে।

MICROPROCESSOR ঃ পার্সোনাল কম্পিউটারের প্রাণ হচ্ছে মাইক্রোপ্রসেসর চিপ। সুতরাং এই যুগ মাইক্রোপ্রসেসরের যুগ। মাইক্রো-প্রসেসর বলতে আমরা বুঝি একটা ইণ্টিগ্রেটেড্ সার্কিট চিপে একটা কম্পিউটারের পুরো কেন্দ্রীয় প্রক্রিয়া বিভাগ। 1971 সালে MCS-4 নামে প্রথম মাইক্রোপ্রসেসর চিপ তৈরি করে আমেরিকার ইণ্টেল কর্পোরেশন। 4 বিট (B1T) শব্দ দৈর্ঘ্যের এই মাইক্রোপ্রসেসর চিপটিতে প্রায় 2250টি ট্রানজিস্টার ছিল। এর পর অত্যন্ত দ্রুত গতিতে মাইক্রোপ্রসেসর চিপের উন্নতি হয়। 8 বিট ও 16 বিট-এর পর এখন 32 বিট-এর মাইক্রোপ্রসেসরও সহজলভ্য। একটা ছোট কম্পিউটারের মধ্যে মাত্র কয়েকটা মাইক্রোপ্রসেসর থাকলেই চলে। তার একটা হল সি. পি. ইউ, একটা ডিস্ক ড্রাইভ নিয়ন্ত্রণ করে, একটা থাকে ভি. ডি. ইউ. টার্মিনালে, একটা কী-বোর্ডে এবং আর একটা প্রিণ্টারে। একটা সাধারণ মাইক্রোপ্রসেসরে থাকে রেজিস্টার, কণ্ট্রোল ইউনিট, অ্যারিথমেটিক অ্যাণ্ড লজিক ইউনিট আর ডেটা ও অ্যাড্রেস বাস।

MODEM ঃ এই ছোট্ট ইলেকট্রনিক যন্ত্রের সাহায্যে দুটো কম্পিউটার টেলিফোন লাইনের মাধ্যমে একে অপরের সঙ্গে তথ্য বিনিময় করতে পারে। মোডেম (MODEM) কথাটা এসেছে মডিউলেটর (MOdulator) এবং ডিমডিউলেটর (DEModulator) শব্দ দুটির সন্ধি থেকে। মোডেমের মারফৎ যোগাযোগের সময়ে কম্পিউটারের ডিজিটাল আউটপুট ঢোকে মোডেমে—মোডেম তাকে অ্যানালগ শব্দ-সঙ্কেতে রূপান্তরিত করে। এই সঙ্কেত টেলিফোনের তার বেয়ে চলে যায় দূরবর্তী আর

এক কম্পিউটারের মোডেমে। সেখানে এই অ্যানালগ সঙ্কেত পুনরায় ডিজিটাল সঙ্কেতে রূপান্তরিত হয়—তখন শুরু হয় দুটো কম্পিউটারের হ্যাণ্ডশেকিং এবং চলে তথ্য বিনিময়।

MULTIPROGRAMMING : সাধারণত কোনও প্রোগ্রামই কম্পিউটারের পুরো স্মৃতিভাণ্ডারকে দখল করে রাখে না। ফলে ওই প্রোগ্রামের যখন কাজ হয় তখন বাকি স্মৃতিভাণ্ডার অব্যবহৃত থাকে। এছাড়া কোনও একটি প্রোগ্রাম যখন ইনপুট-আউটপুট ব্যবস্থার সঙ্গে কাজ করে তখন সি. পি. ইউ.-এর গণনা বিভাগের ক্ষমতা কাজে লাগানো হয় না। এই সব কারণেই দুটি বা তিনটি প্রোগ্রাম একই সঙ্গে কম্পিউটারের স্মৃতিভাণ্ডারে রেখে একসঙ্গে চালানোর পরিকল্পনা করা হয়। এর ফলে বিভিন্ন প্রোগ্রামের নির্দেশ অনুযায়ী সি. পি. ইউ. বিভিন্ন কাজ একই সময়ে করতে পারে। এইভাবে একটি প্রোগ্রাম যখন ইনপুট ব্যবস্থা থেকে তথ্য পাঠ করছে, দ্বিতীয় একটি প্রোগ্রাম তখন গণনার কাজ করছে আর তৃতীয় কোনও প্রোগ্রাম তখন আউটপুট ব্যবস্থায় তথ্য পাঠাতে পারছে। একেই বলে মালটিপ্রোগ্রামিং বা মালটিটাস্কিং (MULTITASKING)।

NETWORK : সময় বিভাজন বা টাইম শেয়ারিং-এর সাহায্যে একটি বড় কম্পিউটারের সঙ্গে অনেক টার্মিনাল লাগানো যায় এবং আমরা একসঙ্গে অনেক কাজ করতে পারি। এক্ষেত্রে টার্মিনাল অনেক দূরেও অবস্থিত হতে পারে এবং কম্পিউটারের সঙ্গে টেলিফোন লাইন, মাইক্রোওয়েভ লিঙ্ক, এমন কি কৃত্রিম উপগ্রহ সংযোগ ব্যবস্থার সাহায্যেও যোগাযোগ করা যেতে পারে। এই ভাবে গড়ে ওঠে কম্পিউটার নেটওয়ার্ক—অর্থাৎ একটি কেন্দ্রীয় কম্পিউটারের সঙ্গে যোগাযোগের অজস্র গতিপথ থাকতে পারে। সাধারণত দু'রকমের নেটওয়ার্ক হয়—স্টার নেটওয়ার্ক এবং লুপ নেটওয়ার্ক। ব্যাঙ্কে, বিমান সার্ভিসে, রেলওয়ে সার্ভিসে এই ধরনের ব্যবস্থা খুবই প্রয়োজনীয়। বিদেশে কোনও ব্যাঙ্কের প্রধান কার্যালয়ের কেন্দ্রীয় কম্পিউটারের নেটওয়ার্কে তার বিভিন্ন শাখা-অফিসগুলি যুক্ত থাকে। তখন শাখা-অফিসে বসেই তথ্য আদান-প্রদানের মাধ্যমে যাবতীয় কাজকর্ম করা যায়। বিমানে বা রেলে আসন সংরক্ষণ ব্যবস্থাতে নেটওয়ার্ক যথেষ্ট কার্যকরী।



প্রমথেশ দাস। জন্ম ১৯৪৮ সালে, কলকাতায়। সাম্মানিক পদার্থবিদ্যায় প্রেসিডেন্সি কলেজ থেকে স্নাতক হন ১৯৬৫ তে। ১৯৬৮ সালে কলকাতা বিশ্ববিদ্যালয়ের ফলিত পদার্থবিদ্যা বিভাগ থেকে এম টেক (বিশ্ববিদ্যালয় রৌপ্য পদক)। বর্তমানে ডেভালাপমেন্ট কনসালট্যান্টস লিমিটেড নামক ডিজাইন ও কারিগরী সংস্থার ইন্সট্রুমেন্টেশন বিভাগের ম্যানেজার। এর আগে ফার্টিলাইজার কর্পোরেশন অফ ইন্ডিয়া এবং হিন্দুস্থান পেপার কর্পোরেশন-এ যুক্ত ছিলেন। স্বদেশে ও বিদেশে প্রযুক্তিবিদ্যার নানা কাজে লেখকের কম্পিউটার ব্যবহারের অভিজ্ঞতা বেশ কয়েক বছরের। 'কম্পিউটার্স ইন ক্লাসরুমস্' নামক ইংরেজি পাঠ্য পুস্তকের লেখক শ্রীদাস ইন্সট্রুমেন্ট সোসাইটি অফ আমেরিকার সিনিয়র মেম্বার এবং ইন্সটিটিউশন অফ ইঞ্জিনিয়ার্স (ইন্ডিয়া)-র মেম্বার।



অনীশ দেব। জন্ম ১৯৫১ সালে কলকাতায়। সাম্মানিক পদার্থবিদ্যায় কলকাতা বিশ্ববিদ্যালয়ের স্নাতক। পরে একই বিশ্ববিদ্যালয়ের ফলিত পদার্থবিদ্যা বিভাগ থেকে ১৯৭৪ সালে বি. টেক (বিশ্ববিদ্যালয় রৌপ্যপদক) এবং ১৯৭৬-এ এম. টেক (বিশ্ববিদ্যালয় স্বর্ণপদক এবং অধ্যাপক পি. এন. ঘোষ স্বর্ণপদক)। কর্মজীবনে ডিজাইন ইঞ্জিনিয়ার হিসেবে তিন বছরেরও বেশি যুক্ত ছিলেন বর্তমান ডেভালাপমেন্ট কনসালট্যান্টস লিমিটেডের ইন্সট্রুমেন্টেশন বিভাগে। পরে ১৯৮৩ থেকে কলকাতা বিশ্ববিদ্যালয়ের ফলিত পদার্থবিদ্যা বিভাগে অধ্যাপনায় যোগ দেন। বিজ্ঞান বিষয়ে বিভিন্ন পত্রপত্রিকায় লেখকের লেখালেখি প্রায় এক দশক ধরে। তাঁর প্রকাশিত বিজ্ঞানের বই 'বিজ্ঞানের হরেকরকম' ও 'বিজ্ঞানের দশ দিগন্ত'। শ্রীদেব ইন্সটিটিউশন অফ ইঞ্জিনিয়ার্স (ইন্ডিয়া)-র মেম্বার।